Herramientas de Segmentación y Evaluación de Series Temporales Basadas en Modelos Ocultos de Markov

Proyecto de Fin de Carrera
Ingeniería Informática

Gonzalo Pérez Verdú
Director Jesús García Herrero

12 de octubre de 2010

Índice general

Índice de figuras

Índice de cuadros

Agradecimientos

Antes de empezar, deseo mostrar mi agradecimiento a las siguientes personas, por que sin su ayuda ni habría terminado este proyecto, ni posiblemente la carrera:

A mi padre, Amadeo.

A mi madre, María.

A mi hermano, David.

A mis amigos, Edu, Alfred, Lucas, Cris, Nahu, Armentia, Patri, Estrella y Flavia.

A Jose, Alzira y Sara.

A mi tutor, Jesús García (y en especial a su paciencia).

A todos los integrantes de BEST (Carlinhos y resto de Europa), en especial a Guillermo (alias Heavy), Jose Munera (alias P€p€) y Miguel Fernández (alias Mixali).

Y a Boss y a Ossi.

Parte I

Introducción y Objetivos

Capítulo 1

Introducción

La minería de datos es un conjunto de técnicas pertenecientes al campo de la Inteligencia Artificial cada vez más utilizadas en múltiples campos, ya que permite analizar las relaciones entre muchas variables. Sin embargo presenta algunos problemas de tipo práctico relacionados con los grandes volúmenes de datos a analizar.

Por esta razón, se utilizan algoritmos capaces de analizar grandes volúmenes de datos buscando patrones en los mismos en vez de analizarlos exhaustivamente. Estos algoritmos son en muchas ocasiones, difíciles de entrenar debido a su complejidad y a la falta de metodologías que tengan en cuenta todas sus características y no únicamente la precisión que demuestren al resolver un determinado problema en condiciones ideales.

Por esas razones, este proyecto pretende profundizar en las técnicas probabilísticas con memoria, principalmente modelos ocultos de Markov para aplicarlas al análisis de series temporales y más concretamente al análisis de las trayectorias seguidas por aeronaves a lo largo de un intervalo determinado de tiempo y a la vez desarrollar una metodología de evaluación que permita tener en cuenta las particularidades de dichos modelos.

Capítulo 2

Objetivos

Los objetivos de este Proyecto de Fin de Carrera se pueden resumir en dos puntos principales: el primero, el desarrollo de una herramienta basada en Modelos Ocultos de Markov (HMM de ahora en adelante, por sus siglas en inglés: *Hidden Markov Model*) que permita entrenar y utilizar dichos modelos tanto en espacios continuos como en discretos para la segmentación y clasificación de series temporales (IV en la página 95); el segundo busca obtener una herramienta que permita evaluar, de un modo lo más objetivo posible, que dicha clasificación es correcta y además pueda aplicarse a otros clasificadores.

Por lo tanto, dividiremos nuestros objetivos en dos puntos principales y los explicaremos por separado.

2.1. Problemas de clasificación de series temporales

Un problema muy habitual en minería de datos es el consistente en clasificar el comportamiento de una evento en una serie temporal en base a una variable (5.1.3 en la página 30).

A modo de ejemplo y para comprobar el funcionamiento de un clasificador sobre una serie temporal compleja, utilizaremos una serie temporal que modela el comportamiento de una aeronave a lo largo de un determinado intervalo de tiempo, mayor o menor dependiendo de la serie en cuestión, de forma que permite representar la trayectoria de la misma en dicho intervalo.

Si bien en dichas series de datos también existía información acerca de otras variables, incluyendo velocidades y aceleraciones, se ha decidido centrarnos en los datos referentes a las posiciones de la aeronave por haber sido obtenidos directamente desde sensores y no calculados o interpolados a partir de otros datos.

Dichas trayectorias presentan ciertas características que las hacen aptas para intentar clasificarlas con determinadas técnicas con memoria: generalmente, los cambios de rumbo son suaves por lo que intentar clasificarlos directamente con otras técnicas, como los árboles 4.5, induce a errores de clasificación en dichas trayectorias y también en aquellas con cambios de rumbo muy suaves o con cambios constantes debidos a correcciones de rumbo pero que en lugar de estar maniobrando, simplemente intentan mantener una determinada trayectoria.

Finalmente, existen determinadas fases de vuelo que son muy estables, como la mostrada a continuación:

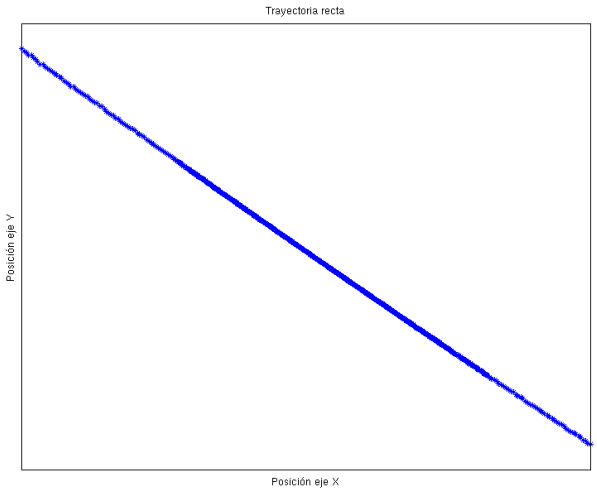


Figura 2.1: Trayectoria recta

En otras trayectorias en cambio, el avión realiza maniobras constantes (hipódromos como los mostrados a continuación):

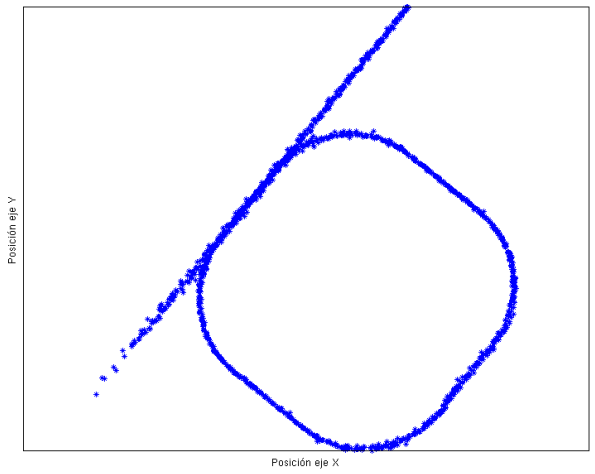


Figura 2.2: Hipódromo

Debido a estas características, se ha buscado un clasificador que parece ser capaz de resolver muchos de estos problemas: los modelos ocultos de Markov (*HMM*), que, como se verá en el epígrafe 7 en la página 45, al tener una cierta

memoria, permite reconocer patrones más complejos que los de otros clasificadores, de forma que nos permite clasificar de forma más acertada las áreas en las que una aeronave se encuentra maniobrando, aunque dicha memoria resulte en ocasiones problemática (ver 8.2 en la página 83).

Así, uno de los objetivos de este proyecto de fin de carrera será desarrollar una herramienta genérica que permita entrenar modelos ocultos de Markov y aplicarla al problema descrito en el epígrafe 6 en la página 41, de forma que permita clasificar las zonas en las que una aeronave está maniobrando de las que se encuentra en vuelo estable.

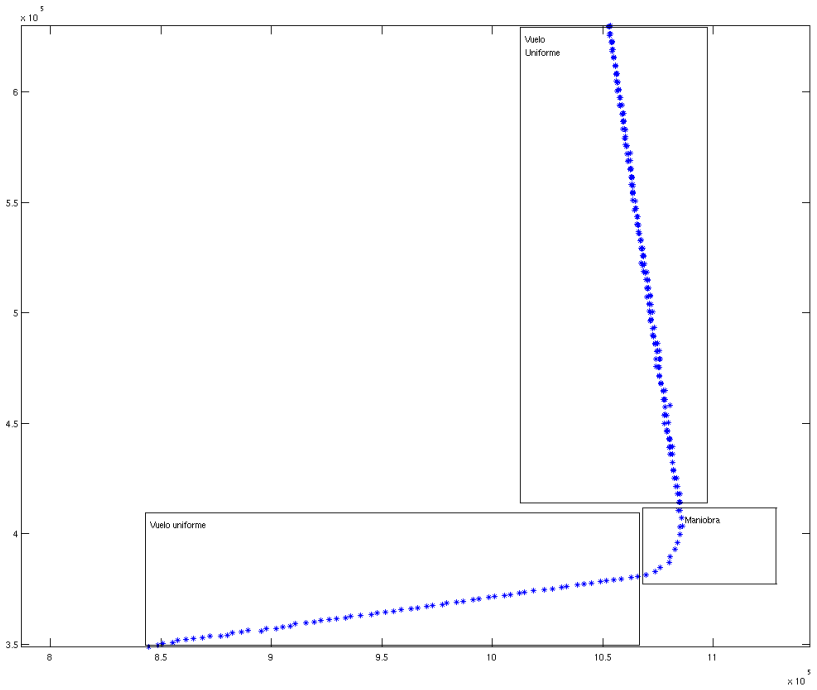


Figura 2.3: Clasificación de trayectorias según se encuentren maniobrando o no

2.2. Evaluación de sistemas clasificadores

Uno de los problemas que apareció en el desarrollo de la herramienta de clasificación que se verá en el epígrafe 10 en la página 97 es que, si bien existen metodologías para la evaluación de herramientas de clasificación, estas son de poca utilidad cuando se evalúan los modelos ocultos de Markov debido a ciertas características de los mismos, más detalladas en el punto 8.2 en la página 83.

Dado que los modelos ocultos de Markov tienen memoria, también tienen una cierta "inercia" a la hora de clasificar correctamente una serie temporal de forma que el uso de evaluadores que funcionan en otros clasificadores, como las matrices de confusión y otros indicados en el epígrafe 8 en la página 76 devuelve resultados incorrectos, clasificando los errores como retrasos y otros problemas

descritos en el punto 8.2 en la página 83.

Debido a dichos problemas, como segundo objetivo de este proyecto de fin de carrera, será desarrollar una metodología de evaluación de clasificadores de series temporales que tenga en cuenta dichas características, así como una herramienta que permita utilizar la metodología en cuestión para evaluar el funcionamiento de los modelos ocultos de Markov frente a otros evaluadores comúnmente utilizados.

Capítulo 3

Organización de la memoria

Para facilitar tanto su lectura como para permitir que el lector pueda saltarse puntos que ya conozca, esta memoria ha sido organizada en cuatro secciones, correspondientes a:

I. Introducción y Objetivos: pretende resumir el ámbito del proyecto y presentar una introducción al mismo.

II. Estado del arte e introducción a técnicas de clasificación: busca introducir los problemas habituales del campo de la minería de datos y series temporales, así como explicar el funcionamiento de los clasificadores y las metodologías a utilizar.

III. Evaluadores: explica la razón de ser y el desarrollo de una metodología de evaluación que tenga en cuenta las características de las técnicas de clasificación a utilizar.

IV. Desarrollo y experimentación de la herramienta de clasificación: explica el funcionamiento de la herramienta de clasificación basada en HMM presentando resultados frente a otros clasificadores, vía la herramienta de clasificación presentada en en la parte 10 en la página 97.

Parte II

Estado del Arte e Introducción a Técnicas de Clasificación

Capítulo 4

Series temporales

Como ya se ha explicado en el epígrafe 2 en la página 11, el objetivo de este proyecto de fin de carrera es el de centrarse en el estudio de algoritmos para el análisis de series temporales y en el desarrollo de una metodología que permita comparar diversos algoritmos, algunos de los cuales tienen ciertas características que hacen complicado evaluarlos vía metodologías más tradicionales (ver 8 en la página 76).

Antes de centrarnos en los posibles algoritmos a analizar y los ejemplos sobre los que los utilizaremos, trataremos de introducir la problemática relacionada con el análisis de series temporales.

Una serie temporal es un conjunto de observaciones tomadas en determinados momentos del tiempo, ordenados cronológicamente y, normalmente, espaciados entre sí de manera uniforme de forma que permite analizar la evolución de un fenómeno o variable a lo largo del tiempo.

El objetivo del análisis de una serie temporal, de la que se dispone de datos en períodos regulares de tiempo, es el conocimiento de los patrones de comportamiento de la variable analizada para prever la evolución futura de la misma, suponiendo que las condiciones que explican su comportamiento no cambiarán respecto a las actuales y pasadas. Esto es: se suele partir de la base de que situaciones pasadas explican situaciones futuras, aunque esto no siempre tiene por que ser cierto.

En el caso más simple, el de los fenómenos deterministas, es posible predecir su comportamiento futuro sin ningún tipo de error, pero en general, las series de interés dependen además de fenómenos aleatorios, de forma que el estudio de su comportamiento pasado sólo permite acercarse a la estructura o modelo probabilístico para la predicción del futuro comportamiento de dicha fenómeno.

Estos modelos se denominan también procesos estocásticos. Así, un proceso estocástico es una sucesión de variables aleatorias X_t, con $t = 1, 2, 3 \ldots n$, que evolucionan con el tiempo (subíndice t). Cuando se dispone de n datos de un proceso estocástico, se está frente a n muestras, de tamaño unidad, extraídas de la población analizada (variable aleatoria), correspondientes al tiempo en que se realizó la medición: este conjunto de datos es la llamada serie temporal o cronológica.

Dado que en dichos procesos dependemos de variables cuyo comportamiento es aleatorio (desde ruido hasta sistemas deterministas inestables), podríamos obtener un modelo que explique el comportamiento de la serie en el período

estudiado, pero a la vez es muy arriesgado el uso de dicho modelo para hacer previsiones a medio o largo plazo, debido a que dicho ruido o la propia naturaleza del evento a modelar, hacen que los resultados teóricos y reales sean muy distintos: por ejemplo, si bien es posible modelizar un fractal de Mandelbrot, definido por una fórmula bastante sencilla, no es posible calcular su evolución futura debido a su naturaleza caótica (4)

Las razones que explican dichos comportamientos son varias:

- No es posible tener en cuenta todas las variables que influyen sobre un sistema: por definición un modelo es una simplificación de la realidad reducida a un grupo de variables que nos permita explicar un comportamiento de un sistema en el tiempo. Sin embargo en muchas ocasiones, nos encontramos ante sistemas que dependen de muchas variables, como en el caso de los modelos meteorológicos.

- Es posible que el sistema sea muy sensible al ruido, como la atracción magnética entre dos imanes, en el cual una variación de la distancia produce respuestas muy diferentes en las salidas:

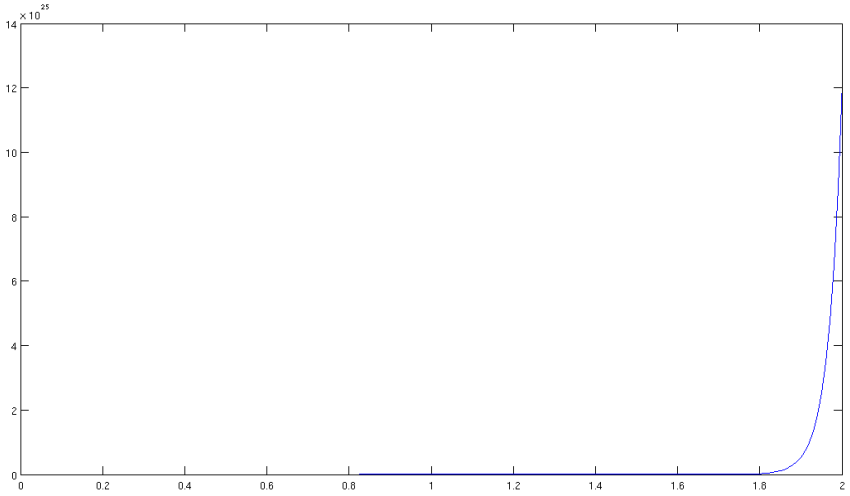


Figura 4.1: Atracción magnética en función de la distancia

- Finalmente, tenemos el caso de sistemas que, aun siendo perfectamente modelables, no podemos conocer su comportamiento debido a que son de naturaleza caótico determinista. Por ejemplo, un atractor de Lorentz es un sistema fácil de modelar: se conocen sus ecuaciones perfectamente, siendo estas dependientes únicamente de tres variables, σ,ρ y β, que siendo constantes dan como resultado un sistema inestable del que nunca llegamos a conocer el comportamiento futuro:

$$\frac{dx}{dt} = \sigma(y - x) \tag{4.1}$$
$$\frac{dx}{dt} = x(\rho - z) - y \tag{4.2}$$
$$\frac{dx}{dt} = xy - \beta x \tag{4.3}$$

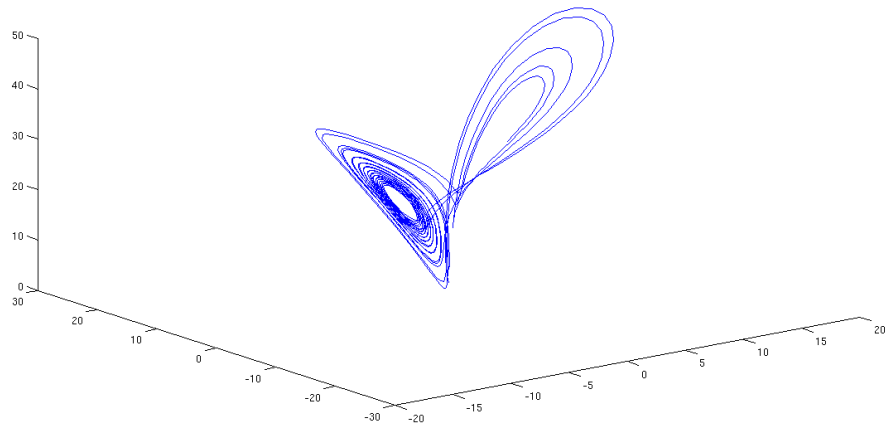


Figura 4.2: Atractor de lorenz

Así, en todas las series cronológicas, debemos tener cuidado con las previsiones a causa de la más que probable inestabilidad del modelo en un futuro más o menos alejado del último instante del que se conocen datos.

Dada la forma en que está representada la información disponible de una serie cronológica, con n muestras de tamaño unidad procedentes de otras tantas poblaciones de distribución y características desconocidas, hacen que las técnicas de inferencia estadística, usualmente aplicadas en muestras de tamaño superior a la unidad, no sean válidas para las series temporales, de forma que es necesario el uso de determinados criterios metodológicos que permitan el estudio de estos fenómenos, y en particular la previsión de su evolución futura, para aplicarlos a los campos que corresponda.

4.1. Análisis de series temporales

Para estudiar el comportamiento de cualquier serie temporal y predecir los valores que puede tomar en un futuro, existen varias metodologías que introduciremos y revisaremos brevemente. Las más comúnmente utilizadas son:

- Análisis clásico, o modelización por componentes, y
- El enfoque Box-Jenkins.

4.1.1. Modelización por componentes

La primera técnica que discutiremos y también en la que centraremos los experimentos relativos al análisis de evaluadores es la del análisis clásico o modelización por componentes.

Este método consiste en identificar, en una serie Y_t, cuatro componentes teóricos, que pueden o no estar presentes en una serie dada:

1. Tendencia, T_t: es la componente general a largo plazo y suele expresarse como una función del tiempo de tipo polinómico o logarítmico, por ejemplo $T_t = \alpha^0 + \alpha^1 t + \alpha^2 t + \ldots$

   Como ejemplo de serie con tendencia tenemos la serie histórica de la demanda eléctrica en España mostrada en la figura 4.8 en la página 26.

2. Estacionalidad, E_t: Las variaciones estacionales son oscilaciones que se producen y repiten en períodos de tiempo cortos. Pueden estar asociadas a factores dinámicos, cuya evolución está claramente ligada a una variable dada.

   Un ejemplo de estacionalidad lo vemos al comparar la serie histórica de la demanda eléctrica en España (figura 4.7 en la página 25) y eliminarle la tendencia (figura 4.8 en la página 26), de forma que obtenemos una estacionalidad como la mostrada en la figura 4.9 en la página 27

3. Ciclos, C_t: Son variaciones similares a la tendencia, pero las variaciones cíclicas se producen a largo plazo y suelen ser más difíciles de identificar cuanto más largo es su período, debido, fundamentalmente, a que el tiempo de recogida de información no aporta suficientes datos, por lo que a veces quedarán confundidas con las otras componentes.

4. Residuos, R_t: La componente residual es la que recoge la aportación aleatoria de cualquier fenómeno sujeto al azar.

   Si a la figura 4.9 en la página 27 se le eliminaran las funciones que rigen la demanda estacional, obtendríamos unos residuos debidos a los diversos fenómenos aleatorios que rigen la demanda eléctrica.

   En el caso del análisis de trayectorias (ver 6 en la página 41), utilizaremos las componentes residuales calculadas a partir de las mismas como entrada para nuestro clasificador ya que el trabajo con dichos residuos resulta más simple que el trabajo con coordenadas.

Para evaluar las distintas componentes se utilizan determinadas técnicas estadísticas tales como: modelado lineal, medias móviles, diferencias finitas, etc. Dado que no se pretende una presentación exhaustiva de técnicas de análisis, no se profundizará en las mismas pero se introducirán brevemente (ver epígrafe 4.2 en la página 24) a fin de que este trabajo sea lo más auto-contenido posible.

Así, y suponiendo que el componente aleatorio (residuo) es aditivo, podemos identificar el resto de componentes de forma que se combinen tendencia, estacionalidad y ciclos para dar lugar a la serie definitiva.

A fin de combinar dichos componentes, existen dos modelos comúnmente utilizados: los denominados aditivos y multiplicativos.

- Modelo aditivo: $Y = T + E + C + R$

- Modelo multiplicativo: $Y = T{\cdot}E{\cdot}C + R$.

Para una primera identificación visual de la serie en cuestión, podemos considerar que si el patrón estacional se mantiene con amplitud constante se tratará de modelo aditivo, como por ejemplo la siguiente serie dada por la ecuación $f(T) = seno(T) + T$:

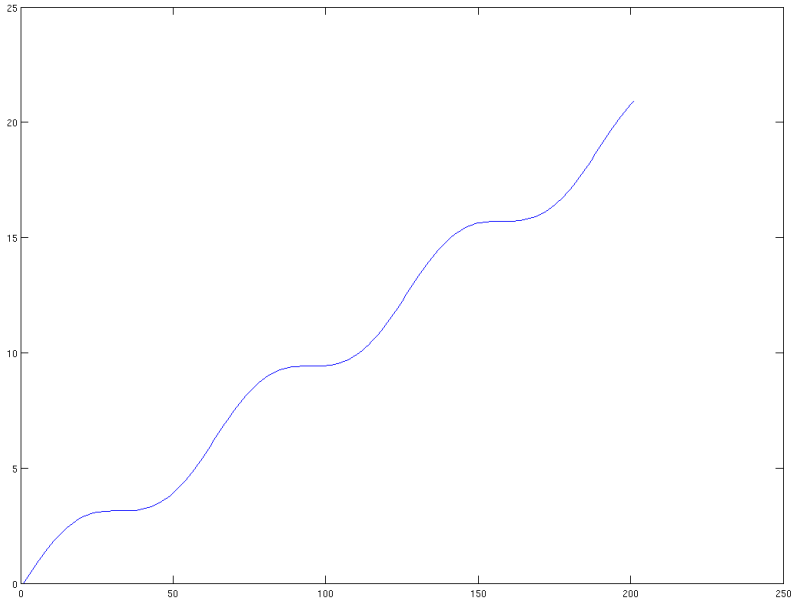


Figura 4.3: Modelo aditivo

En este modelo aditivo se puede interpretar que la estacionalidad se da con una tendencia es de tipo lineal, y que la estacionalidad es de período $p = 2\pi$, es decir, cada 4 unidades de tiempo se repite el patrón. En este tipo de modelos, la estación componente del período da lugar a una recta con ordenada en el origen distinta para cada caso y pendiente común a todos.

En el caso de nuestro ejemplo, si obtenemos sus componentes, tendremos las siguientes gráficas:

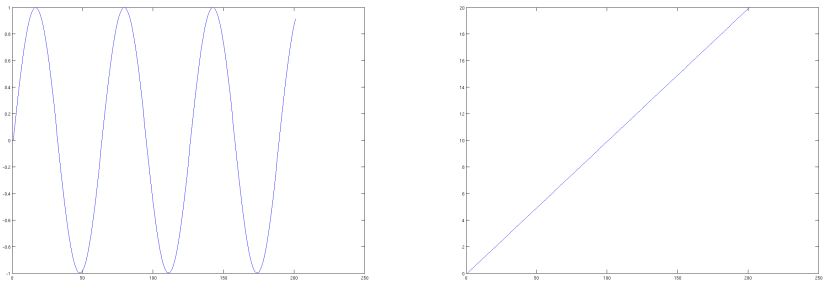


Figura 4.4: Componentes del modelo aditivo

En cambio, cuando dicho patrón se vaya amplificando con el tiempo, será multiplicativo, como el de la siguiente figura definida por la ecuación $f(T) = sin(T) \cdot T$, formado por los mismos componentes (4.4) que el modelo aditivo mostrado en 4.3 en la página anterior:

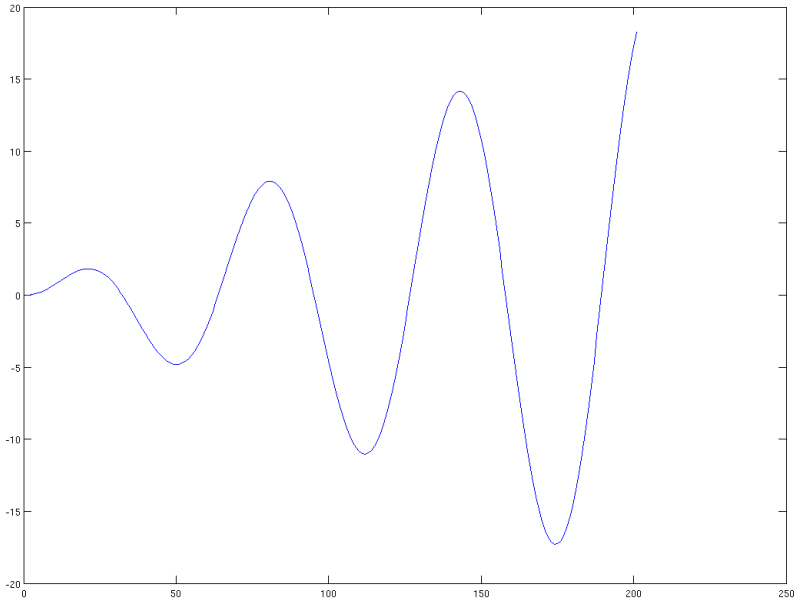


Figura 4.5: Modelo multiplicativo

En el modelo multiplicativo, el componente estacional actúa sobre la ordenada en el origen y sobre la pendiente; prescindiendo de los ciclos y suponiendo una tendencia lineal tipo $T_t = \alpha_0 + \alpha_1 t$ y una estacionalidad de período p, para cualquier $t = p + s$, con $s = 1, \ldots, p$, cada una de las p estaciones del período configura una recta distinta, tanto en lo que se refiere a la ordenada en el origen ($\gamma_0 s$) como a la pendiente ($\gamma_1 s$). El conjunto de las p rectas constituye el modelo

de comportamiento de la serie.

En cualquier caso, esta división en modelos estrictamente aditivos o estrictamente multiplicativos es bastante restrictiva, por lo que es mejor utilizar un modelo mixto mucho más general para que sea posible tener en cuenta casos como el mostrado en la siguiente figura (figura 4.6) que al estar representado por la ecuación $f(T) = seno(T)\cdot \mathrm{T}+\mathrm{T}$ resulta ser muy difícil de analizar por cualquiera de los modelos aditivos o multiplicativos puros.

$\sigma_1 = 8,493$ y $\sigma_2 = 3,3582$, con unas probabilidades de emisión dadas por $\mu_1 = 0,8493$ y $\mu_2 = 1,8168$,

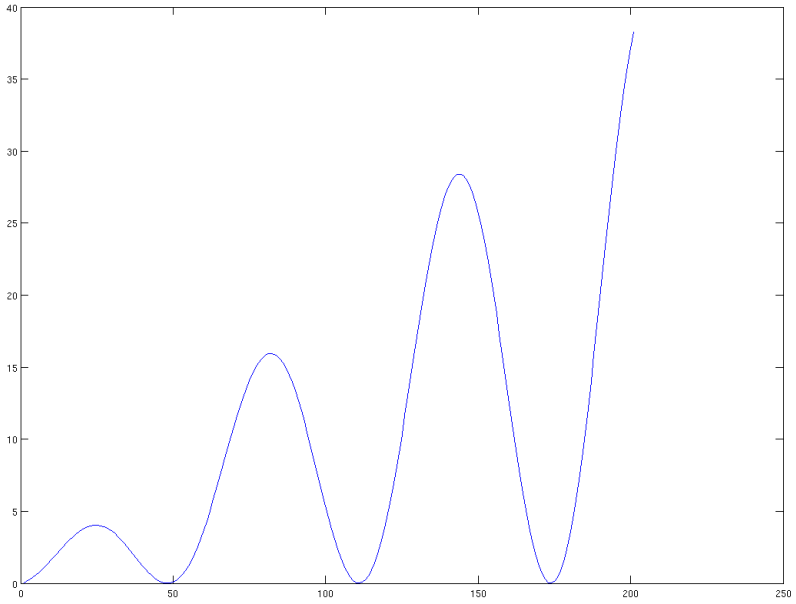


Figura 4.6: Modelo mixto

4.1.2. Enfoque Box-Jenkings

Otro posible enfoque es el de la metodología de Box-Jenkins, que dirige el esfuerzo a determinar cuál es el modelo probabilístico que rige el comportamiento del fenómeno a lo largo del tiempo. Es decir, partiendo de la premisa de que no siempre va a ser posible identificar los componentes de la serie, se trata de estudiar el componente aleatorio puro, reflejado en los residuos. La metodología estadística utilizada en el estudio de una serie temporal por este sistema, se basa en los siguientes pasos:

1. Identificación del modelo.
2. Estimación de los parámetros.
3. Validación de los supuestos admitidos en el análisis, (diagnosis del modelo).

Para poder abordar esta metodología es imprescindible, en primer lugar, estudiar un conjunto de modelos de comportamiento que cubran el mayor espectro posible de los procesos estocásticos objeto de interés.

Dado que los modelos con los que vamos a trabajar son probabilísticos (ver 7), este enfoque es en principio el más correcto para el problema que nos ocupa, sin embargo también se tendrán en cuenta ciertas características de la modelización por componentes que se explicarán en 4.2.

4.2. Análisis de series temporales

Habiendo explicado los posibles enfoques, nos centraremos en la modelización por componentes, utilizando ambos modelos, cada uno para tratar el problema al que mejor se adaptan:

- Para el caso del análisis puro de trayectorias, se utilizó la modelización por componentes a fin de reducir el nivel de ruido de la misma.
- Para el desarrollo de los modelos que expliquen el comportamiento de nuestro sistema (6), y en general para el desarrollo de los clasificadores probabilísticos (7 y 10) se utiliza el enfoque Box-Jenkings.

4.2.1. Descomposición de series temporales

Con este método, también denominado sistema clásico, se descompone la serie en tendencia, estacionalidad, ciclos y residuos. Una vez decidida la conjunción entre ellos, aditiva o multiplicativa, se obtiene el modelo con el que hacer previsiones.

La tendencia es la componente más importante de la serie, al definir lo que se podría interpretar como comportamiento a largo plazo. Cada observación va ligada a un valor del tiempo, lo que permite plantear un modelo del tipo

$$Y = \varphi(t) + \varepsilon \tag{4.4}$$

Donde la función $\varphi(t)$ puede ser:

- Lineal: $\varphi(t) = \alpha_0 + \alpha_1 t$
- Polinómica: $\varphi(t) = \alpha_0 + \alpha_1 t + \alpha_2 t^2 + ... + \alpha_n t^n$
- Exponencial: $\varphi(t) = \alpha_0 t^{\alpha_1}$

Si la serie no presenta estacionalidad, el método de estimación mínimo-cuadrática y todas las pruebas de hipótesis relativas a la explicación del modelo y a la significación de los coeficientes estimados, propios del modelo lineal ordinario, permiten estimar los coeficientes del modelo de tendencia sobre los datos directos. Caso de existir componente estacional, para que ésta no enmascare la tendencia, es necesario estabilizar previamente la serie; a modo de ejemplo, sea la demanda eléctrica mensual en España desde enero de 1959 a Enero de 2008:

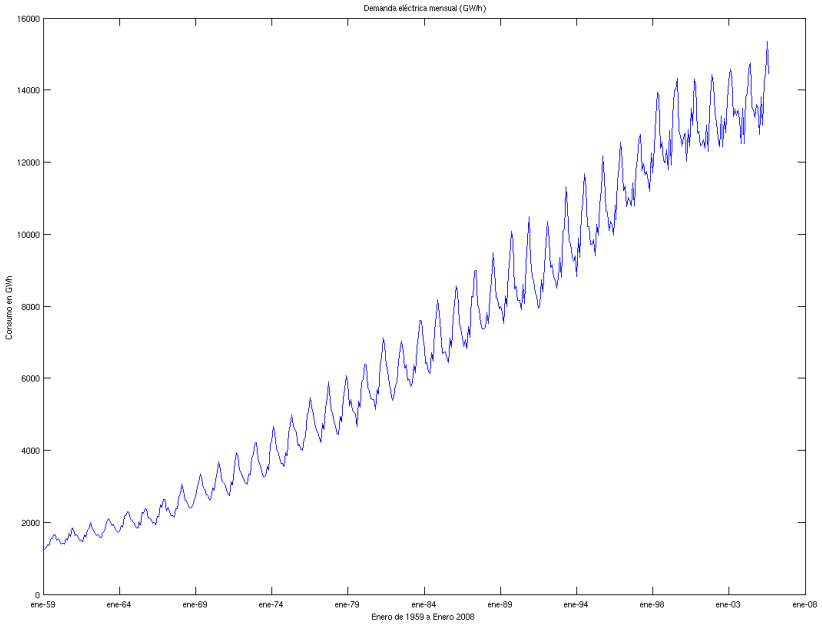


Figura 4.7: Demanda mensual de energía eléctrica (GWh) en España, Enero de 1959 a Enero de 2008

Se ha elegido esta serie por que se puede observar que la misma tiene una tendencia creciente y una tendencia estacional. Si nos interesara obtener la tendencia estacional, por ejemplo para clasificar los meses con mayor consumo deberíamos eliminar primero la tendencia de la serie, para lo cual habría que calcularla, en este caso mediante una aproximación por mínimos cuadrados (similar a la que se utilizará en el caso de las trayectorias a lo largo de este documento):

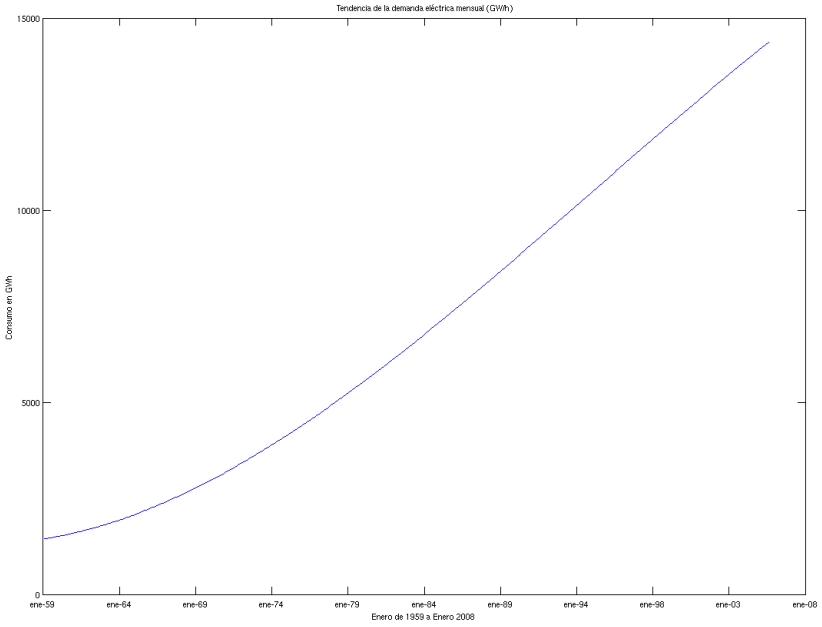


Figura 4.8: Tendencia del consumo de la demanda eléctrica en España, Enero de 1959 a Enero de 2008

Una vez obtenida la tendencia de la serie, es posible eliminarla de los datos de la serie principal y calcular cuál es la variación en el consumo de la demanda eléctrica a lo largo de los meses del año según avanza el tiempo. En este caso se observa que si bien a lo largo de la serie histórica la demanda ha sido más alta en los meses de invierno, en años recientes ha crecido también en los de verano.

Conociendo estos datos y disponiendo de un modelo que explique el comportamiento de la demanda, que en años anteriores se aproximaba a un $seno(x)$ que hacía que cada doce meses hubiera un ciclo con máximo en invierno y que en años recientes ha duplicado su frecuencia $seno(2x)$, podemos intentar ofrecer hipótesis que expliquen el fenómeno, como por ejemplo que en invierno la demanda eléctrica crece por el frío y la falta de luz y que en años recientes, también crece en verano debido a los aires acondicionados:

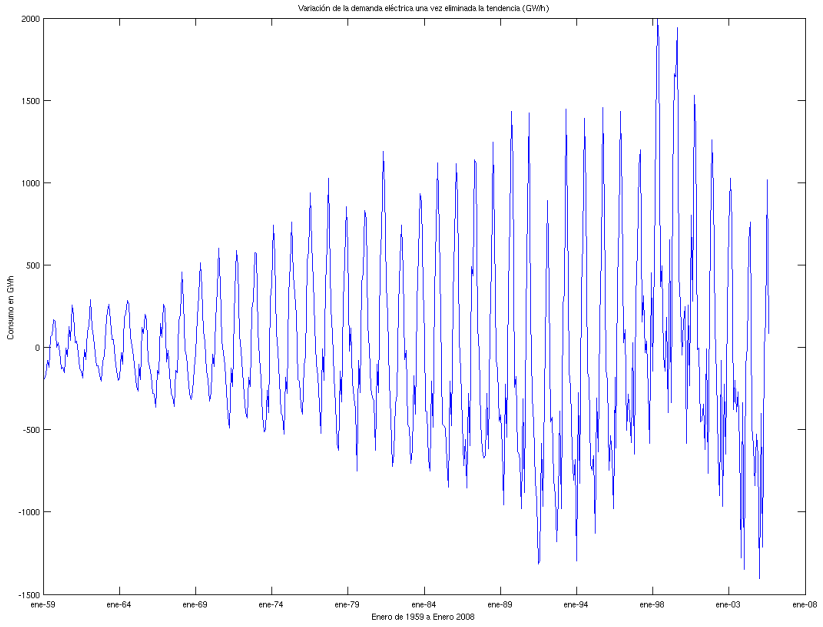


Figura 4.9: Variación mensual de la demanda de energía eléctrica en España, Enero de 1959 a Enero de 2008

Capítulo 5

Introducción a la minería de datos

El análisis de series temporales es una posible aplicación de la minería de datos. La minería de datos intenta extraer información no trivial de los datos, a fin de utilizar dicha información para algún proceso. Dicho de otro modo, la minería de datos prepara, sondea y explora los datos para sacar la información oculta en ellos.

Mediante los modelos extraídos utilizando técnicas de minería de datos se aborda la solución a los problemas de *predicción*, *clasificación* y *segmentación*.

Aunque el problema que nos interesa y que intentaremos resolver es el de la clasificación y segmentación de series temporales en unos grupos conocidos, se describen brevemente los problemas relativos a la minería de datos:

- *Clasificación:* consiste en examinar las características de nuevas entidades descubiertas en los datos y asignarle una clase predefinida.
- *Segmentación:* tiene como objetivo el segmentar a un grupo diverso en un conjunto de subgrupos. A diferencia de clasificación, no depende de clases predefinidas.
- *Predicción:* vista con un enfoque científico, la predicción es una declaración, cuanto más precisa mejor, de lo que puede ocurrir si se cumplen unas condiciones especificadas. Esto es, tras el análisis de los datos, se extraen patrones de comportamiento que permiten predecir ciertos fenómenos caso de cumplirse determinadas condiciones.

En nuestro caso nos centraremos en el análisis de algoritmos aplicados a problemas de clasificación y la evaluación de dichos algoritmos, aplicados a trayectorias de vuelo con clases predefinidas.

5.1. Problemas típicos de minería de datos

La minería de datos intenta ajustar modelos o determinar patrones de comportamiento a partir de datos, debiendo tener en cuenta las limitaciones del trabajo con modelos estadísticos: principalmente que los modelos son, por su

propia definición, incompletos, con lo cual se debe trabajar con un determinado margen de error y que además, cuando se trabaja con datos provenientes del mundo real, suele existir una cantidad dada de ruido con la que es necesario contar.

Como se ha comentado en el punto anterior, los algoritmos de minería de datos realizan en general tareas de clasificación (descripción de datos y patrones a partir de clases predefinidas de datos), de predicción (a partir de de datos desconocidos, intenta predecir las los fenómenos que ocurrirán después) y de segmentación de datos (descripción de datos y patrones sin clases predefinidas).

Existen además otras técnicas, como análisis de dependencias e identificación de anomalías se pueden utilizar tanto para descripción como para predicción, dado que este proyecto se centra en la descripción de herramientas de clasificación y de evaluación de dichos clasificadores, nos limitaremos a realizar un pequeño resumen de los principales problemas encontrados:

5.1.1. Descripción

La descripción consiste en realizar un análisis preliminar de los datos (características de los datos, casos extremos, etc.) a fin de conseguir que el usuario se sensibilice con los datos. La descripción de los datos trata de obtener características concisas de los mismos (vg.: medias, desviaciones estándares).

5.1.2. Segmentación

La segmentación busca dividir los datos en grupos (también llamados clústeres) de elementos que tienen propiedades similares. Por ejemplo se puede segmentar dos poblaciones en base a sus distancias a dos puntos centrales:

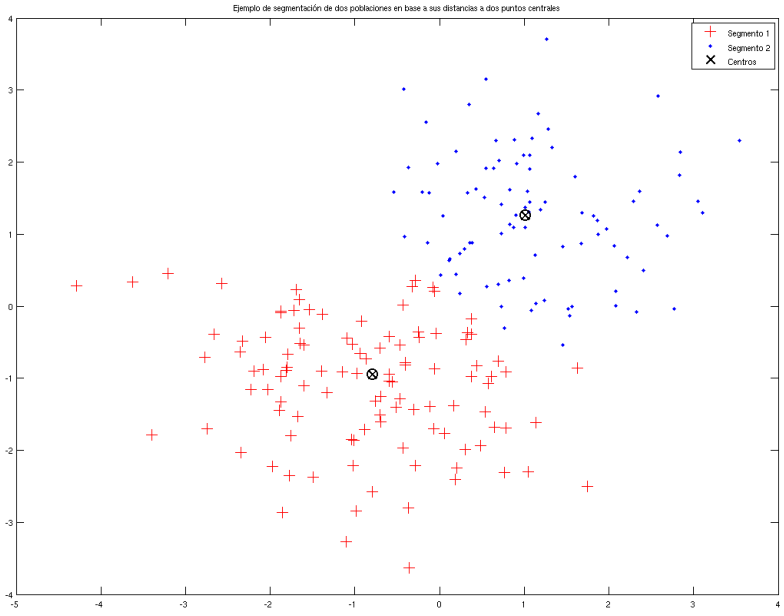


Figura 5.1: Ejemplo de la segmentación de dos poblaciones en función de su distancia a dos puntos centrales

5.1.3. Clasificación

La *clasificación,* es un forma de predicción que utiliza un modelo basado en observaciones pasadas para predecir observaciones futuras. Consiste en examinar las características de nuevas entidades descubiertas en los datos y asignarle una clase predefinida.

Desde el punto de vista de la clasificación, los datos aparecen como objetos que están caracterizados por ciertos atributos, pertenecientes a distintas clases predefinidas (y con valores discretos). Así, la clasificación busca obtener un modelo que permita predecir a qué clase pertenecen los datos dados los valores de dichos atributos.

A modo de ejemplo, dada la siguiente trayectoria de una aeronave (se muestran las distintas coordenadas que ha ido ocupando en un plano):

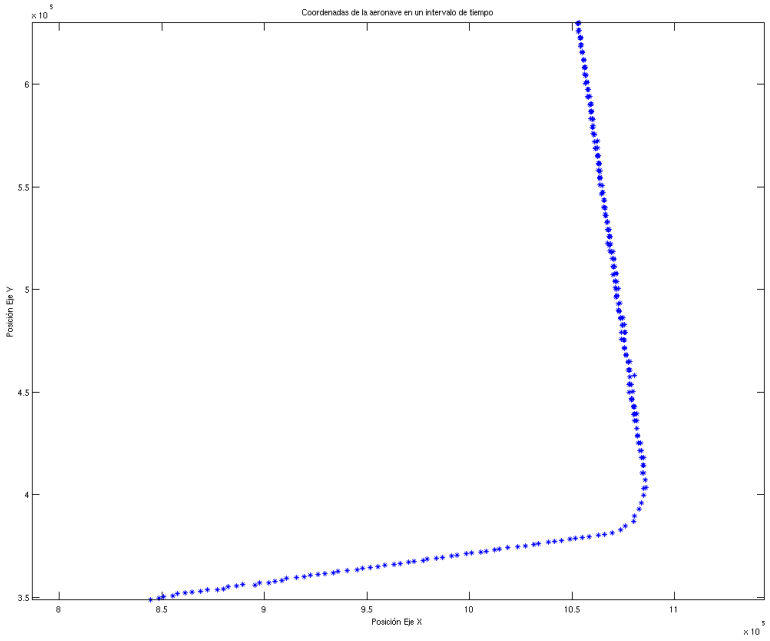


Figura 5.2: Coordenadas de una aeronave en un intervalo

Deseamos clasificarla en dos posibles clases de datos:

1. Clase de datos de 'Vuelo uniforme': maniobra únicamente para hacer pequeñas correcciones de rumbo.

   a) Clase de datos 'Maniobra': realiza cambios de rumbo

   El resultado de dicha clasificación sería el que se muestra a continuación:

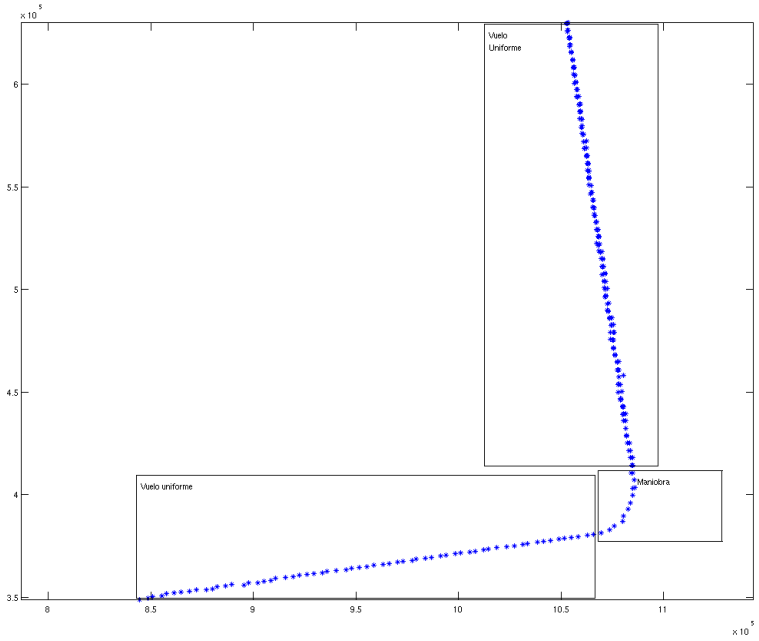


Figura 5.3: Clasificación del comportamiento de una aeronave en un intervalo

Además de la clasificación propiamente dicha, hay que tener en cuenta una serie de técnicas estrechamente relacionadas con ésta:

- La *estimación*, busca la misma meta que la clasificación, pero aplicada a clases continuas, trata de obtener un modelo para poder predecir el valor de la clase, dados los valores de los atributos.
- *Análisis de dependencias:* Utiliza el valor de un elemento para predecir el valor de otro. La dependencia puede ser probabilística, puede definir una red de dependencias o puede ser funcional (e.g. leyes físicas).
- *Detección de desviaciones, casos extremos o anomalías:* Detecta los cambios más significativos en los datos con respecto a valores pasados o normales. Sirve para filtrar grandes volúmenes de datos que probablemente pueden ser menos interesantes. El problema en estos casos está en determinar cuándo una desviación es significativa para ser de interés.

5.2. Clasificadores comúnmente utilizados

Como ya se ha comentado antes, la clasificación consiste en la asignación de los datos en categorías basadas en un atributo previsible. Cada dato contiene un conjunto de atributos, uno de los cuales es el atributo clase (atributo predecible). La clasificación busca encontrar un modelo que describa el atributo clase en función de atributos de entrada. Así, para obtener un modelo de clasificación,

es necesario disponer del valor de la clase de los casos de entrada en el conjunto de datos de formación, que suelen ser datos históricos.

Algunos de los algoritmos de clasificación comúnmente utilizados en minería de datos son:

- Árboles de decisión: modelan funciones discretas, en las que el objetivo es determinar el valor combinado de un conjunto de variables, y basándose en el valor de cada una de ellas, determinan la acción a tomar. Los ejemplos más utilizados (y frente a los cuales validaremos nuestro clasificador) son los árboles ID3 y C4.5. Son útiles para clasificación y predicción, pero funcionan mejor en problemas de pequeño tamaño y con pocas variables.

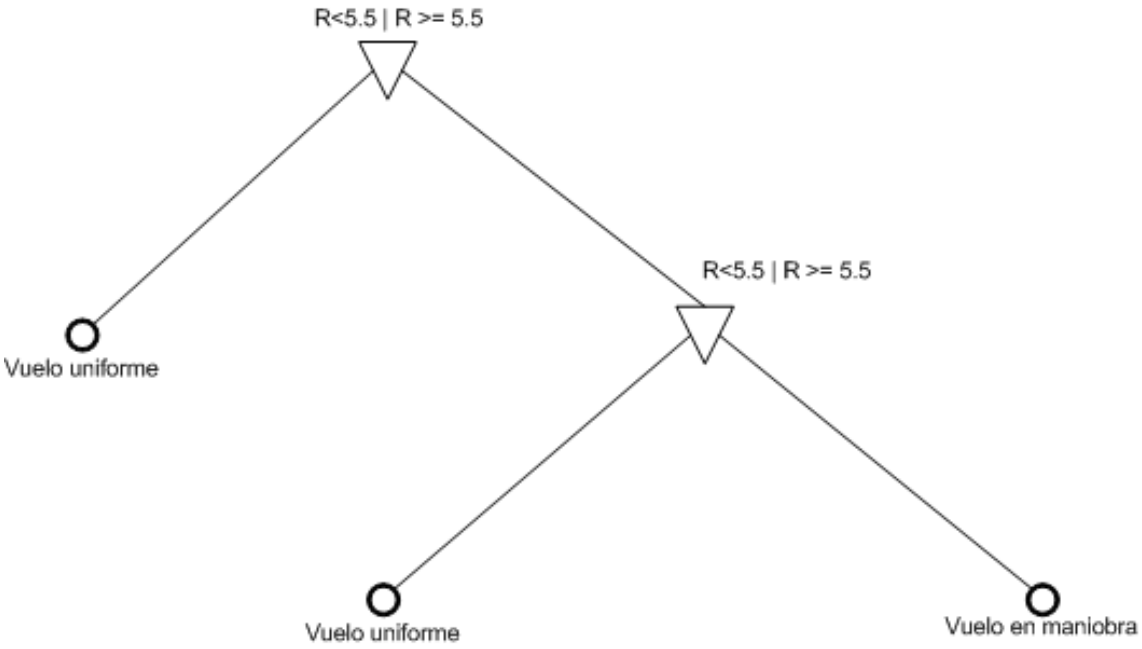


Figura 5.4: Ejemplo de árbol de decisión

- Redes de neuronas: son conjuntos de neuronas artificiales (cada neurona es una simplificación del funcionamiento de una neurona natural, tienen un conjunto de entradas que procesan y envían señales de salida a través de conexiones), por sí mismas son limitadas pero se usan en conjuntos (divididos a su vez en capas) para dividir las señales a procesar y reconocerlas. Son efectivas en la clasificación, predicción y segmentación de datos, pero su entrenamiento es complejo y requieren de muchos recursos.

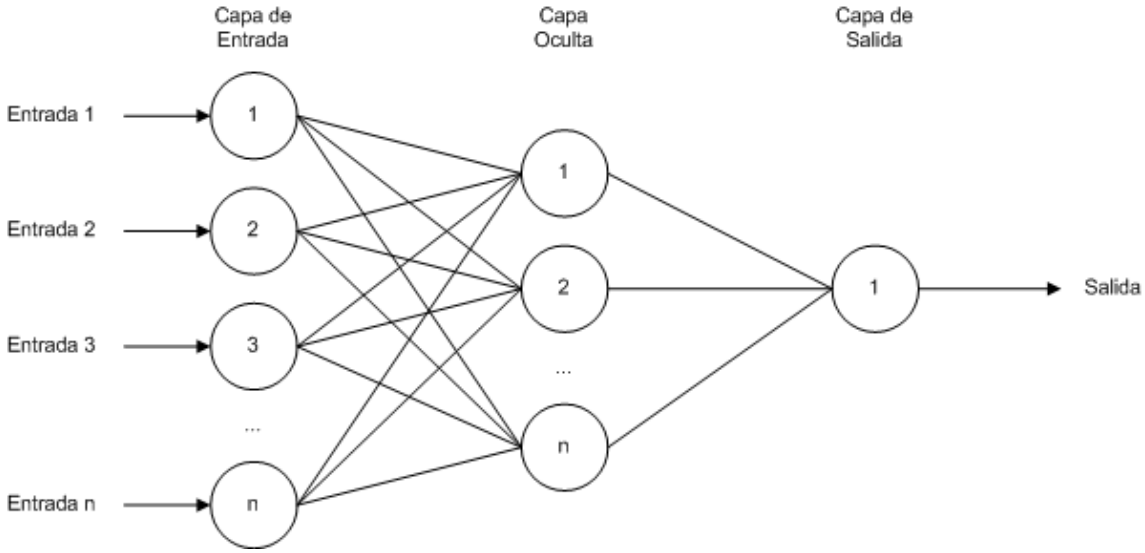


Figura 5.5: Ejemplo de red de neuronas

- Redes bayesianas: son una representaciones compactas de una función de probabilidad conjunta. Formalmente, una red Bayesiana es un grafo acíclico dirigido, en el que cada nodo representa una variable aleatoria y las relaciones de dependencias e independencias condicionales quedan establecidas en la propia estructura de la red.

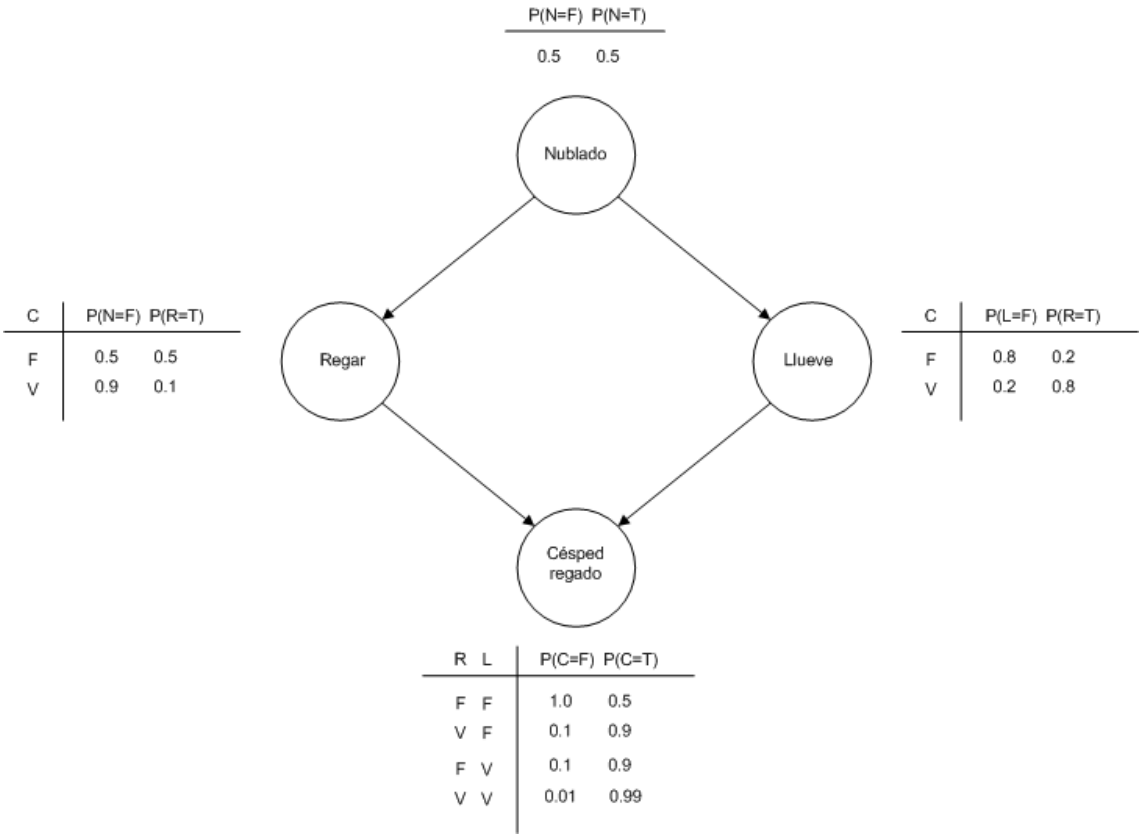


Figura 5.6: Ejemplo de red bayesiana

Los modelos ocultos de Markov , con los cuales se ha construido el clasificador que describiremos más adelante, son uno de los ejemplos más simples de una red Bayesiana.

Una de las razones por la que se ha elegido, el comportamiento de los Modelos Ocultos de Markov [1] es que, a diferencia de otras redes bayesianas más complejas, se conocen algoritmos de entrenamiento (7.5.1) y para clasificar los estados por las que ha ido pasando el HMM (Viterbi, 7.4)

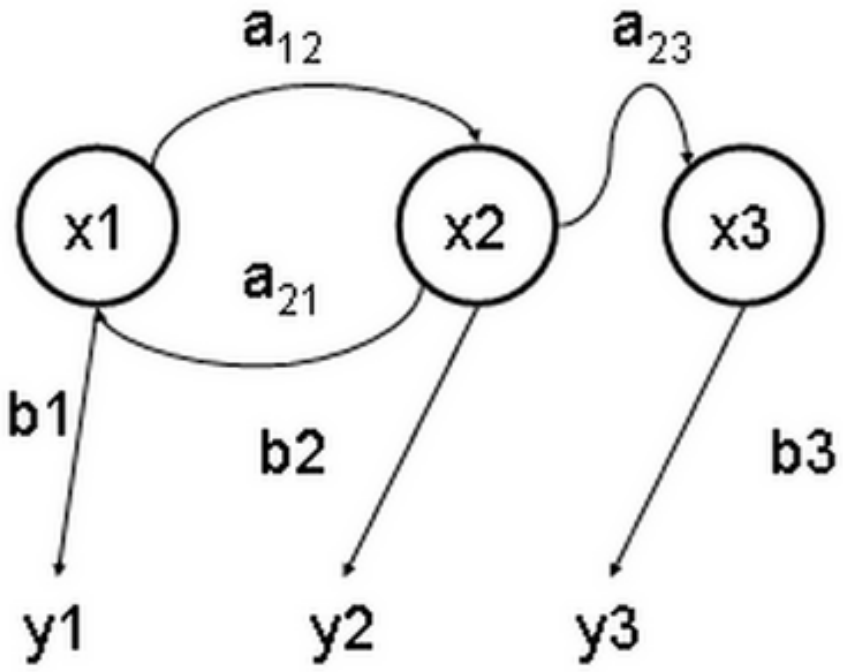


Figura 5.7: Ejemplo de Modelo Oculto de Markov

5.3. Tratamiento de series temporales en minería de datos

El tratamiento de una serie temporal para extraer datos de la misma depende de qué modelo siga la fuente de los datos: en muchos casos puede bastar con una media móvil auto-regresiva (ARMA, [2]), basada en distribuciones normales, en cambio, si queremos calcular la probabilidad de que un determinado número de eventos ocurra un intervalo, será necesario utilizar distribuciones de poisson [3].

5.3.1. Mezclas de modelos

Las mezclas de modelos se utilizan para representar funciones de densidad de probabilidad complejas, a partir de la marginalización de distribuciones conjuntas entre variables observadas y variables ocultas. Se utilizan en la clasificación y segmentación de series temporales en las que se describen comportamientos complejos, para las cuales es necesario utilizar mezclas independientes de modelos, de forma que cada una de las características a analizar en dicha serie temporal venga dada por una distribución. Por ejemplo puede darse el caso de que no podamos calcular con una única distribución normal el comportamiento de una serie por estar formada por dos fenómenos, el primero modelizado por una normal $N(0,1)$ y el segundo por una $N(1,2)$ como se muestra en la figura siguiente:

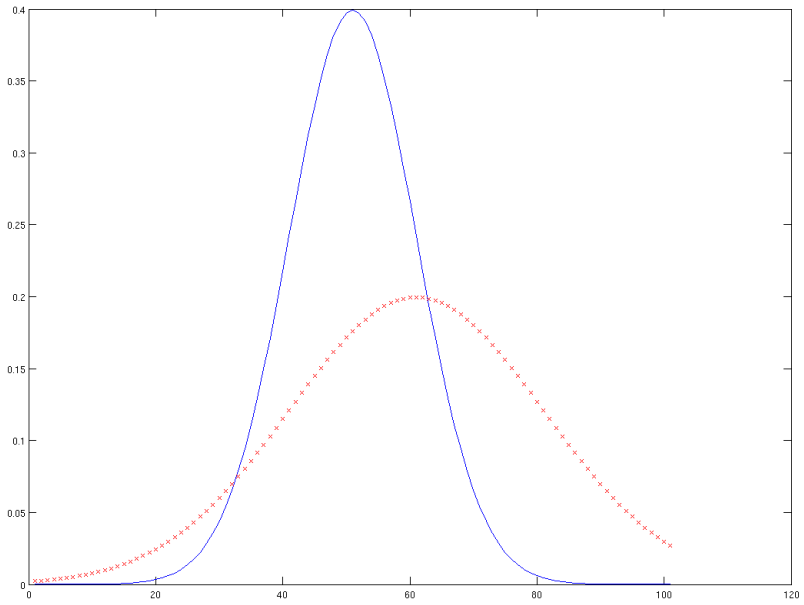


Figura 5.8: Mezcla de gaussianas

Así, de forma general, las mezclas de modelos son modelos probabilísticos de estimación de densidad calculados a partir de mezclas de distribuciones, utilizados para clasificar subgrupos de atributos en los datos o para representar una distribución no normal.

Una mezcla de modelos como las utilizadas por nuestro algoritmo de clasificación (ver 10) está formada por la mezcla de dos distribuciones normales (gaussianas) utilizada para representar las probabilidades de maniobra de una aeronave en una serie temporal, como la que se muestra a continuación:

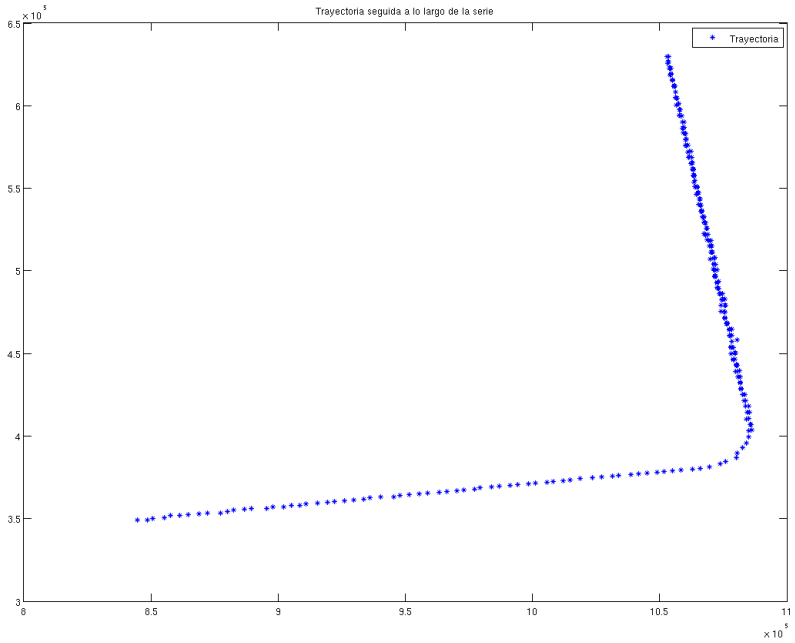


Figura 5.9: Serie a clasificar

En el ejemplo mostrado, no es posible representar las zonas de maniobra de la aeronave sólo con una distribución normal dado que existen dos posibles estados en la misma:

- El primero, el del vuelo uniforme: la trayectoria permanece en dicho estado la mayor parte de la serie, la media de la serie en este estado y para esta maniobra es cercana a $0,05$.
- El segundo, el de la maniobra: dura solo unos instantes, pero la media de la misma en esos instantes se acerca a 12

Con valores tan dispares, no resulta posible utilizar una única distribución capaz de representar a ambas, pero en cambio es posible utilizar dos distribuciones normales que permitan clasificar de forma correcta dicha serie temporal.

En el caso de este ejemplo y como se verá en la parte destinada a resultados, más concretamente en 12.2 en la página 114, se utilizan dos distribuciones normales:

$$N_1(0{,}8493,\ 8{,}493) \tag{5.1}$$

$$N_2(1{,}8168,\ 3{,}3582) \tag{5.2}$$

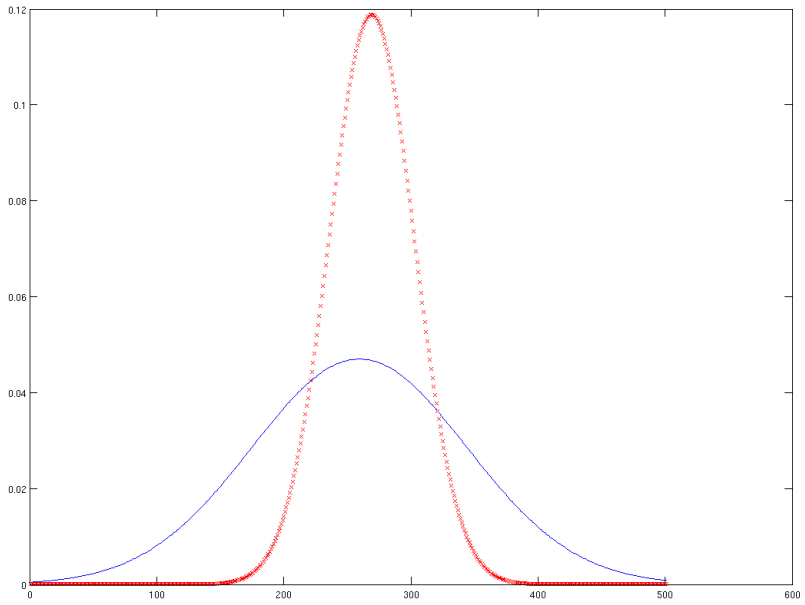


Figura 5.10: Mezcla de gaussianas utilizadas para la clasificación

De esta manera, en vez de intentar representar todas las observaciones con una misma distribución, asignamos a cada categoría de la observación una distribución de probabilidad, de forma que este conjunto de probabilidad determine la probabilidad de que un dato observado venga de una u otra distribución.

Otro posible problema del tratamiento de series temporales resoluble mediante el uso de mezclas de modelos es el que se da cuando los datos muestran dispersiones mucho mayores de lo que se podría esperar.

En estos casos, la mezcla de dos o más distribuciones permite acomodar mejor la presencia de datos heterogéneos que no hubieran sido observados previamente: esto permite que el modelo tenga en cuenta datos no observados aún con distribuciones distintas a las realizadas.

Otra de las ventajas del uso de mezclas de modelos es su flexibilidad: es posible expresar el comportamiento de la serie mediante una distribución normal con una determinada media salvo en determinadas situaciones más extremas en las que nos pueda interesar utilizar otro tipo de distribuciones.

5.3.2. Estimación de parámetros

El mayor problema del uso de mezclas de modelos es el relacionado con la correcta estimación de parámetros, habitualmente se realiza con algoritmos de maximización de la probabilidad (ML) [4]:

$$L(\vartheta_1, ..., \vartheta_m, \delta_1, ..., \delta_1 | x_1, ..., x_n) = j = 1i = \prod_{i=1}^{n} \sum_{i=1}^{m} \delta_i \Pr_i(x_j, \vartheta_i) \tag{5.3}$$

Donde $\vartheta_1, \ldots, \vartheta_m$ son los parámetros de las distribuciones $\delta_1, \ldots, \delta_m$ son los pesos de las mezclas de los modelos (suman 1) y $,x_1, ..., x_n$ las observaciones. Esto implica que para calcular cada parámetro es necesario realizar $2^m - 1$ cálculos, por lo que la estimación de dicha probabilidad se convierte en un problema intratable a pocos parámetros que sea necesario calcular, de manera que no es posible calcular un resultado analíticamente siendo necesario hacer aproximaciones.

Por ejemplo: supongamos que $m = 2$ y que estamos trabajando con una distribución de Poisson con medias λ_1 y λ_2, con unos pesos tales que $\delta_1 + \delta_2 =$ 1,la distribución de la mezcla de parámetros vendría dada por:

$$\Pr(x) = \delta_1 \frac{\lambda_1^x e^{-\lambda_1}}{x!} + \delta_2 \frac{\lambda_2^x e^{-\lambda_2}}{x!} \tag{5.4}$$

Sabiendo que $\delta_2 = 1 - \delta_1$, nos quedan tres parámetros a calcular: λ_1 , λ_2 y δ_1:

$$L(\lambda_1, \lambda_2, \delta_1 | x_1, \ldots, x_n) = \prod_{i=1}^{n} \left(\delta_1 \frac{\lambda_1^x e^{-\lambda_1}}{x!} + (1 - \delta_1) \frac{\lambda_2^x e^{-\lambda_2}}{x!} \right) \tag{5.5}$$

La resolución de este problema de maximización es muy compleja, dado que L es el producto de n factores, cada uno de los cuales es a su vez una suma, así, es más simple tomar dos aproximaciones:

- Maximización numérica de los resultados de la función de probabilidad L o
- El uso del algoritmo EM, que veremos explicado en 7.5.1.

Capítulo 6

Introducción al problema de las trayectorias

Como ejemplo de clasificación de series temporales con modelos ocultos de Markov y como ejemplo de desarrollo de una metodología de evaluación, se utilizará un problema de clasificación de trayectorias de vuelo, con datos tanto simulados como reales (recogidos por Eurocontrol).

Si bien dentro de los ficheros de datos existen múltiples variables como el rumbo, velocidad y aceleraciones, se tomó la decisión de centrarse en las trayectorias y no en las velocidades o aceleraciones: estas son relativas a cada trayectoria y no sirven para clasificar de forma absoluta si la aeronave se encuentra o no en fase maniobra, aunque podrían estudiarse para clasificar otros patrones de las mismas, como podrían ser los tiempos de despegues y aterrizajes u otras operaciones.

Dado que las trayectorias se representa originalmente en el fichero como las coordenadas que ocupa la aeronave en un instante dado, su estudio de forma directa presenta ciertas complicaciones:

- En el caso de trayectorias reales y dependiendo del radar que detecta al avión, se hacen más o menos muestreos por unidad de tiempo: en algunos casos se puede dar una posición por segundo y en otros, una posición cada minuto.
- Los datos están afectados por ruido, en ocasiones no parecen seguir una trayectoria rectilínea debido al efecto del ruido.
- Al estar en términos de coordenadas es difícil trabajar con ellos.

Por estas razones y para simplificar el trabajo de análisis de trayectorias, se utilizó un proceso de muestreo que transforma los datos, seleccionando una pequeña ventana de tiempo uniforme, para obtener un residuo que es el utilizado para comprobar si el avión está o no maniobrando. Una de las principales ventajas del uso de residuos es que permite tratar la información de maniobra del avión con un dato en coma flotante, que es el utilizado para clasificar el comportamiento del mismo.

Así, en casos de trayectorias sin maniobras como la del ejemplo:

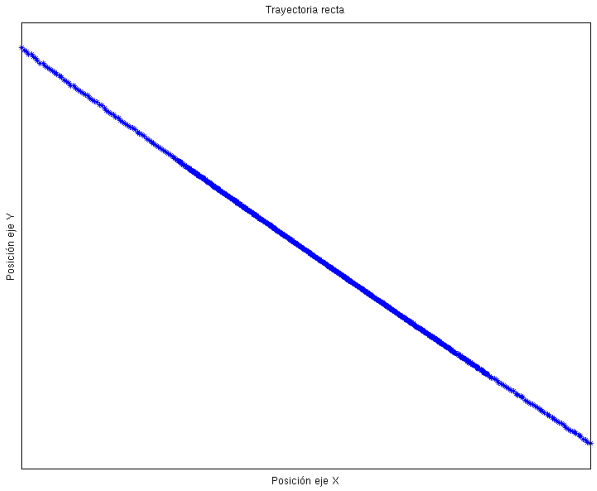


Figura 6.1: Trayectoria recta

A simple vista se observa una trayectoria uniforme y sin ruido, de forma que apenas es necesario realizar ninguna transformación que nos ayude a visualizarla, pero si intentamos realizar el mismo proceso en el siguiente ejemplo:

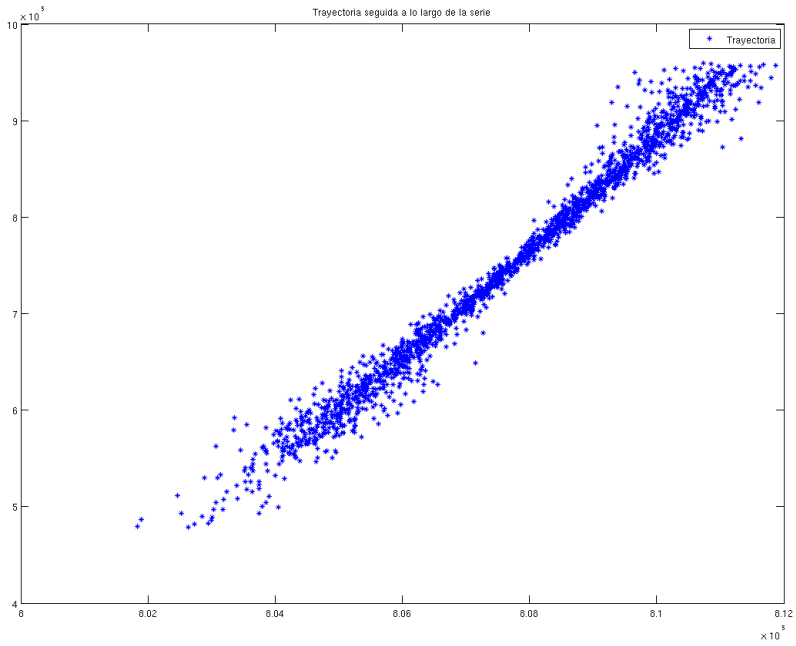


Figura 6.2: Trayectoria con ruido

veremos que mientras la media de toda la trayectoria también parece ser cercana a la mostrada por la trayectoria anterior, sin embargo existe mucha más dispersión en los puntos que la forman. Así, y para reducir en la medida de lo posible los niveles de ruido que se sufren en algunos casos y abstraer el problema de trabajar con coordenadas, se aplica una trasformación para ajustar la trayectoria no a unos puntos si no a una curva. En concreto, se aplica una función de *mínimos cuadrados* [6], que permite, partiendo de un conjunto de pares de puntos, encontrar una función que se aproxime a los datos en base a un criterio de mínimos cuadrados. De esta forma obtenemos para las trayectorias mostradas en las figuras 6.1 en la página anterior y 6.2 en la página anterior los siguientes residuos

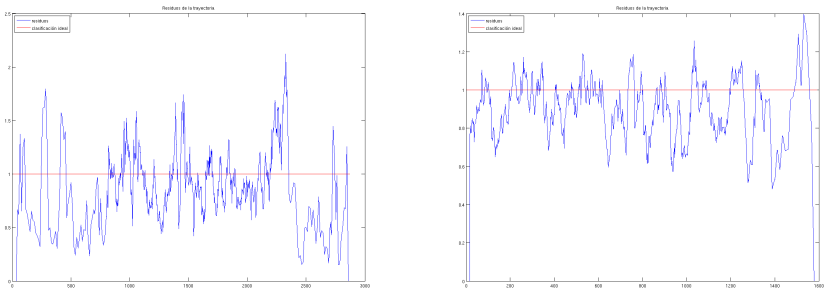


Figura 6.3: Residuos de las trayectorias

que como se puede observar son bastante similares.

Este procedimiento tiene la ventaja de ser óptimo en muchos aspectos (teorema de Gauss-Markov, [6]) y ha sido estudiado en uno de los trabajos en que se basa este proyecto [5].

Ahora bien, cuando se mezclan residuos de trayectorias en estados de vuelo uniforme y de vuelo en maniobra, empiezan a surgir dificultades a la hora de analizar este tipo de series temporales sólo mediante clasificadores comunes, ya que existen trayectorias que, a priori, son fáciles de reconocer pero otras en cambio dependen mucho de los umbrales bajo los que se trabaja:

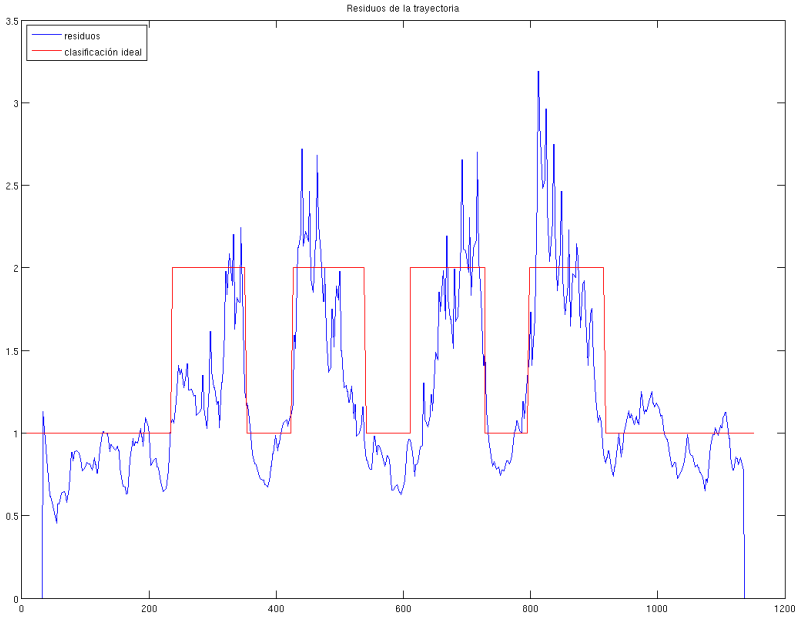


Figura 6.4: Residuos de trayectorias complejas

Dichas trayectorias son difíciles de clasificar, por que el hecho de que el avión se esté moviendo de un lado para otro puede significar o bien que esté maniobrando o que esté realizando correcciones de rumbo debido al mal tiempo; para clasificarlas correctamente es mejor disponer de un clasificador que tenga memoria para tener en cuenta los instantes previos.

Por estas razones, se decidió comprobar el comportamiento de clasificadores bayesianos, en particular el más simple de ellos: los modelos ocultos de Markov, para comprobar si eran capaces de clasificar correctamente las trayectorias y además trabajar en una metodología de evaluación que permita comparar objetivamente unos clasificadores con otros, sean o no bayesianos.

Capítulo 7

Introducción a los Modelos Ocultos de Markov

Antes de explicar el modelado del problema de análisis de trayectorias y los resultados obtenidos y para facilitar la comprensión de los problemas con el uso de modelos ocultos de Markov [1], se explicará brevemente las bases y el funcionamiento de dichos modelos. Para hacerlo, se utilizará un ejemplo más simple que el mencionado de las trayectorias, consistente en clasificar el comportamiento de un determinado dado para saber si está o no trucado.

Si bien hubiéramos podido seguir utilizando dicho ejemplo a lo largo de esta memoria, optamos por el problema de las trayectorias por ser más completo y por modelar un proceso que se da en el mundo real y del cual los datos no dependen de un generador de número pseudoaleatorio como es el caso de los dados.

Una vez explicado el funcionamiento de los Modelos Ocultos de Markov, se presentará una sección dedicada a resultados experimentales, en dicha sección (V en la página 109) los experimentos se centrarán en el análisis de series de datos correspondientes a trayectorias de aeronaves, dichos datos se obtienen a partir de una transformación (ver 6) que a partir de los datos de las distintas posiciones de las aeronaves, genera unos residuos que son los que se utilizarán para clasificar el comportamiento de la trayectoria en base a estados de trayectoria o no trayectoria.

Será con dichos residuos con los cuales trabajaran los clasificadores, empezando por trayectorias muy simples (rectas) y en posteriores pruebas se comprobará la capacidad de clasificar los datos utilizando trayectorias mucho más complejas como las mostradas a continuación:

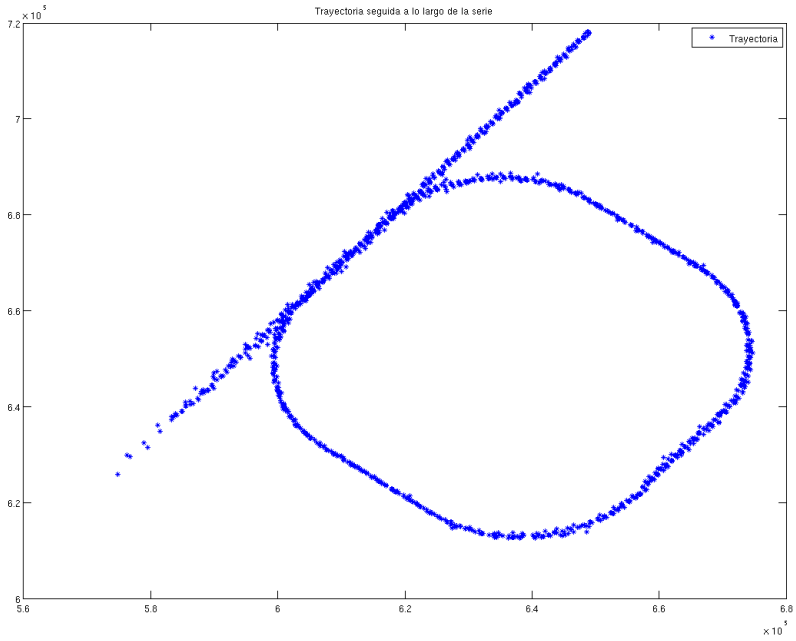


Figura 7.1: Hipódromo

Los clasificadores que se utilizarán serán dos:

- Un árbol binario, por ser un clasificador muy conocido y probado.
- Un modelo oculto de Markov, elegido porque al tener memoria (ver 7.1.2 y [1]), son capaces de reconocer patrones complejos. También son clasificadores muy conocidos y utilizados en determinados campos (reconocimiento de voz y tratamiento de imágenes [1] y [7]).

Además de estos dos modelos, muchas de las trayectorias han sido marcadas a mano de forma que se conoce a priori en que partes de la trayectoria se está maniobrando. Esto se es así para facilitar el entrenamiento de ambos clasificadores y también para facilitar el funcionamiento de la metodología de evaluación. Una vez entrenados ambos modelos, comprobaremos si el modelo oculto de Markov tiene mejor o peor rendimiento que el árbol binario. Así, el árbol binario será utilizado como grupo de control y entrenado con los mismos datos que el modelo oculto de Markov.

Trabajaremos de esta forma porque el árbol binario es un algoritmo muy conocido y probado pero con ciertas desventajas al reconocer patrones en series temporales, mientras que el modelo oculto de Markov tiene ciertas peculiaridades, que veremos más adelante, que hace que no sea posible utilizar sobre él un mecanismo más "clásico" de evaluación, como podrían ser las matrices de confusión.

Así, cada vez que deseemos comprobar la capacidad de ambos algoritmos para clasificar una trayectoria, los ejecutaremos con los mismos datos y evaluaremos sus resultados, tanto con trayectorias completas, como en ciertos casos en

los que se hará trabajar a los clasificadores en pequeñas ventanas de las trayectorias para comprobar si realiza correctamente el análisis de la trayectoria con el cambio de escala y también para comprobar el rendimiento de los clasificadores.

Antes de centrarnos en la metodología de evaluación, intentaremos resumir la motivación y funcionamiento de los modelos matemáticos, en nuestro caso aplicados al problema indicado y el por qué de una metodología de evaluación para compararlos.

Como se ha indicado en la sección dedicada a objetivos (2.2), buscamos obtener una metodología lo más completa posible que nos permita analizar qué capacidad tiene un modelo para clasificar datos en una serie temporal y además que permita comparar varios modelos entre sí. En este punto se ha de puntualizar, para no confundirlo con modelo oculto de Markov, que el modelo será una abstracción de dicho problema a fin de analizarlo, describirlo, explicarlo, predecirlo y simularlo.

Para crear dicho modelo (modelos en realidad, ya que trabajaremos con varios algoritmos con características distintas) plantearemos primero una serie de hipótesis que sean lo bastante completas para representar el comportamiento de los fenómenos a analizar (vuelo de las aeronaves) pero que a la vez sea lo bastante sencillo como para poder manipularlo y estudiarlo. De esta manera, hemos de tener en cuenta que dicho modelo o modelos matemáticos que utilizaremos, no son completamente exactos con los fenómenos que se observarían en la vida real, si no que se trata de una idealización de los mismos. Esto se debe a varias razones, la principal es la imposibilidad de contar con todos los datos que dan lugar a un fenómeno (e.g. el avión puede no estar maniobrando si no corrigiendo su rumbo debido al viento, pero al no tener datos de viento, nuestro modelo llega a la conclusión de que está maniobrando) o que, aún disponiendo de los mismos, tenerlos todos en cuenta haga intratable nuestro problema.

Dado que se trata de idealizaciones, trabajaremos siempre con determinados márgenes de error (intervalos de confianza en 8.1.2) que deberán ser tenidos en cuenta.

Dicho esto, primero explicaremos que modelos existen, con cuales trabajaremos y los pasos a seguir para la creación de un modelo matemático; empezaremos por describir que tipos de modelos existen y cómo podemos clasificarlos:

- Modelos Determinista. Permite conocer de manera puntual la forma del resultado ya que no hay incertidumbre. Además, los datos utilizados para alimentar el modelo son completamente conocidos y determinados.

- Modelos Estocásticos. Son probabilísticos: no se conoce el resultado esperado, sino su probabilidad y existe por tanto incertidumbre. No se conocen todos los datos que alimentan al modelo o no se pueden tener en cuenta todos los datos. Estos son los modelos que utilizaremos para clasificar el comportamiento en vuelo de las aeronaves.

Si además queremos tener en cuenta el origen de la información utilizada para construirlos, podríamos también clasificar a los modelos entre los siguientes tipos:

- Modelos heurísticos Son los que están basados en explicar las causas o mecanismos que dan lugar a un fenómeno a estudiar.

- Modelos empíricos. Son los que utilizan las observaciones directas o los resultados de experimentos del fenómeno estudiado. Dado que no nos centraremos en explicar las causas que explican el comportamiento en vuelo de las aeronaves, por ser estas bien conocidas y además por no tener todos los datos que las expliquen, utilizaremos los datos de que disponemos para crear un modelo empírico que clasifique su comportamiento.

Finalmente, y centrándonos en nuestra metodología de evaluación de modelos, tendremos en cuenta que determinados modelos matemáticos encuentran distintas denominaciones dependiendo de sus aplicaciones finales, según lo cual un modelo puede ser:

- Modelos conceptuales. Son los que, utilizando formulas y/o algoritmos, reproducen los procesos físicos que se producen en la naturaleza.
- Modelo matemático de optimización. Los modelos matemáticos de optimización se utilizan en diversas ramas de la ingeniería para resolver problemas que por su naturaleza son indeterminados por presentar más de una solución posible. En concreto, nuestra metodología de evaluación corresponde a uno de estos modelos, dado que dependiendo de los datos y de otras variables, es posible que en unos casos un clasificador se comporte mejor que otro, pero que en el cómputo global ambos generen resultados similares. Por esta razón nos interesa contar con un modelo, representado en nuestra metodología de evaluación, que nos permita calcular cual de los modelos a utilizar se comporta mejor en determinadas circunstancias y de forma cuantitativa.

Una vez hemos explicado que tipos de modelos existen, repasaremos brevemente los pasos que hay que seguir para crear un modelo e indicaremos las referencias a los puntos de este documento dónde se llevan a cabo dichos pasos:

1. Encontrar un problema del mundo real. En nuestro caso, buscamos obtener un modelo que nos permita evaluar correctamente el funcionamiento de uno o varios clasificadores, en especial el que utilizamos en 10 en la página 97.
2. Formular un modelo matemático acerca del problema, identificando variables (dependientes e independientes) y estableciendo hipótesis que puedan tratarse de manera matemática, 6 en la página 41 .
3. Aplicar los conocimientos matemáticos que se posee para llegar a conclusiones matemáticas, de forma que se puedan utilizar dichos conocimientos para crear una herramienta capaz de predecir el comportamiento del problema a modelar (10).
4. Comparar los datos obtenidos como predicciones con datos reales, V en la página 109. Si los datos obtenidos de las predicciones son diferentes, se reinicia el proceso para buscar un nuevo modelo, pero al trabajar con modelos probabilísticos, como veremos en 7.1.1 en la página siguiente, trabajamos con una cierta incertidumbre, razón por la cual nos interesa comprobar no el funcionamiento de un modelo frente a los datos reales, si no el de varios modelos entre sí, a fin de compararlos para ver cual de ellos funciona mejor o en que casos funciona mejor, 8 en la página 76.

Debemos tener en cuenta, respecto a este punto, que los modelos de cualquier clase pueden ser poco prácticos si no están respaldados con datos confiables. Si se distorsionan las estimaciones, la solución obtenida, aún siendo más o menos óptima en un sentido matemático, en realidad será de calidad inferior desde la perspectiva del sistema real y por tanto de poco uso.

Por esa razón la disponibilidad de datos tiene un efecto directo en la precisión del modelo, de ahí la elección de dos clasificadores y el uso como datos de entrada series temporales con los datos de vuelo de diversas aeronaves: se dispone de gran disponibilidad de datos por esa parte y dichos datos varían lo suficiente como para analizar el comportamiento de los clasificadores en múltiples circunstancias, como ante trayectorias fáciles de clasificar, como la descrita en 12.1 en la página 111 o trayectorias más complejas o con más ruido como la descrita en 12.4 en la página 120.

7.1. Conceptos previos

Antes de seguir es necesario definir algunos conceptos. Concretamente nos centraremos en los dos conceptos que más importancia tienen para comprender los modelos ocultos de Markov [1] y [3]:

1. Proceso estocásticos
2. Propiedad de Markov

7.1.1. Procesos Estocásticos

Un proceso estocástico se define como una colección de variables aleatorias: $\{X_1, X_2...X_t\}$, donde el subíndice t toma valores de un conjunto T dado,

$$0 \leq t \leq T \tag{7.1}$$

usualmente. La variable X representa una característica de interés que se mide en un instante t.

Habitualmente, el estudio del comportamiento de un sistema durante un periodo suele llevar al análisis de un proceso estocástico con la siguiente estructura: en instantes específicos del tiempo t, el sistema se encuentra exactamente en una posición de un número finito de estados mutuamente excluyentes y exhaustivos $0, 1, 2...S$. Los periodos en el tiempo pueden encontrarse a intervalos iguales o su esparcimiento puede depender del comportamiento general del sistema en el que se encuentra sumergido el proceso estocástico.

De esta forma, la representación matemática del sistema físico es la de un proceso estocástico $\{X_i\}$, en donde las variables aleatorias se observan en $t = 1, 2...T$ y en donde cada variable aleatoria puede tomar el valor de cualquiera de los S + 1 enteros $0, 1, 2...S$ que caracterizan los estados del proceso.

7.1.2. Propiedad de Markov

Se dice que un proceso estocástico tiene la propiedad de Markov cuando solo del estado presente se puede obtener información del comportamiento futuro del

proceso, esto es: sus estados futuros son independientes de los estados pasados, mientras que en un proceso estocástico sin la propiedad de Markov, dada una distribución de variables aleatorias $\{X_k, k = 1, 2...n\}$, la probabilidad de que una variable aleatoria X_j esté en el estado x_j es $P[X_j = x_j|\{X_k\}, k \neq j]$. Esto significa que la probabilidad de que dicha variable X_j esté en el estado x_j depende de los valores de todas las demás variables aleatorias $\{X_k\}$.

La propiedad de Markov enuncia que, siendo $\{X(t),\, t \geq 0\}$ un proceso estocástico continuo en el tiempo con valores de t enteros y no negativos, se dice que dicho proceso es un proceso discreto de Markov (cumple la propiedad de Markov) sí, para $n \geq 0$ y en los instantes $0 < t_0 < t_1 < ...t_n < t_{n+1}$ y en los estados $i_0, i_1...i_{n+1}$ cumple que:

$$\Pr(X(t_{n+1}) = i_{n+1}|X(t_n) = i_n,\, X(t_{n-1}) = i_{n-1}, \ldots, X(t_0) = i_o) \tag{7.2}$$

$$\Pr(X(t_{n+1}) = \Pr(X(t_{n+1}) = i_{n+1}|X(t_n = i_n)) \tag{7.3}$$

7.1.3. Intensidad

Se deriva del problema de intentar predecir el la probabilidad de que se de un evento en el instante inmediatamente futuro que se haya producido todavía en un tiempo T:

$$\Pr(T \leq t + \triangle t|T > t) \tag{7.4}$$

Si se desarrolla dicha ecuación mediante la definición de la probabilidad condicionada obtenemos:

$$\Pr(T \leq t + \triangle t|T > t) = \frac{\Pr(T \leq t + \triangle t,\, T > t)}{\Pr(T > t)} = \frac{\Pr(t < T \leq t + \triangle t)}{\Pr(T > t)} \tag{7.5}$$

$$\Pr(T \leq t + \triangle t|T > t) = \frac{F(t + \triangle t) + F(t)}{1 - F(t)} \tag{7.6}$$

Siendo F la función de distribución de T, suponiendo F derivable tendríamos:

$$F(t + \triangle t) - F(t) = f(t)\triangle t + o(\triangle t) \tag{7.7}$$

Así, la probabilidad que deseábamos calcular sería equivalente a $\lambda(t)\cdot\triangle t$, siendo $\lambda(t)$ la llamada función de intensidad para T.

$$\lambda(t) = \frac{f(t)}{1 - F(t)} \tag{7.8}$$

Si T no ocurre en el momento t entonces la probabilidad de que fuera a suceder en el intervalo $(t,\, t + \triangle t]$ para un $\triangle t$ pequeño sería aproximadamente proporcional a dicho $\triangle t$, siendo $\lambda(t)$ dicha constante de proporcionalidad.

7.1.4. Matrices estocásticas

Sirven para representar una cadena de Markov sobre un espacio finito de S elementos. Siendo la probabilidad de cambiar desde el estado i hasta el estado j: $\Pr(j|i)$, representaremos dicha probabilidad en la fila i, elemento j de la matriz: a_{ij}.

$$\Pr = \begin{pmatrix} a_{11} & a_{12} & \dots & a_{1j} & \dots \\ a_{21} & a_{22} & \dots & a_{2j} & \dots \\ & & \ddots & \vdots & \\ a_{i1} & a_{i2} & \dots & a_{ij} & \dots \end{pmatrix} \tag{7.9}$$

Entre las matrices estocásticas nos podemos encontrar con que estas puede ser de los siguientes tipos:

- Matriz estocástica a derecha, es la que se utiliza en procesos de Markov, es cuadrada, con todos sus elementos no negativos y sus filas suman 1:
  - $\forall i, j\, a_{ij} \geq 0$
  - $\sum_j a_{ij} = 1$
- Matriz estocástica a izquierda, , es cuadrada, con todos sus elementos no negativos y sus columnas suman 1:
  - $\forall i, j\, a_{ij} \geq 0$
  - $\sum_i a_{ij} = 1$
- Matriz estocástica doble o bi-estocástica, es cuadrada, con todos sus elementos no negativos y sus filas y columnas suman 1:
  - $\forall i, j\, a_{ij} \geq 0$
  - $\sum_j a_{ij} = 1$
  - $\sum_i a_{ij} = 1$

7.1.5. Cadenas de Markov

Las cadenas de Markov son procesos estocásticos en tiempo discreto que cumplen con la propiedad de Markov, esto es, un sistema que puede describirse como estando en uno de los posibles estados del un conjunto de N estados $X_1, X_2, \dots X_N$ y cambiando cada cierto tiempo de estado de acuerdo con un conjunto dado de probabilidades asociado al estado actual.

Si llamamos al instante actual n, pudiendo ser $n = 1, 2 \dots$ y el estado actual en el instante n a X_n y sabiendo que el sistema cumple la propiedad de Markov, para calcular el siguiente estado no será necesario el calculo completo del estado actual y todos los estados pasados, si no que basta con que tengamos la información del estado actual y del previo:

$$\Pr(X_{n+1} = x | X_n = x_n,\, X_{n-1} = x_{n-1} \dots,\, X_1 = x_1) = \Pr(X_{n+1} = x | X_n = x_n) \tag{7.10}$$

Dado que en este modelo la salida del proceso es el conjunto de estados, cada uno de los cuales corresponde a un posible evento físico observable, podemos denominarlo como modelo observable de Markov, a diferencia de un modelo oculto de Markov, que es el que veremos más adelante.

Como resultado de esta propiedad, se puede decir que los procesos de Markov tienen memoria, concretamente dependen de su estado inmediatamente anterior.

7.1.6. Propiedades

Algunas de las principales propiedades de los Modelos ocultos de Markov son las que se enumeran a continuación.

7.1.6.1. Cambio al estado siguiente

La probabilidad de cambiar de un estado i al inmediatamente posterior j y denotada por p_{ij} es:

$$p_{ij} = \Pr(X_1 = j | X_0 = i) \tag{7.11}$$

7.1.6.2. Cambio a nuevo estado

La probabilidad de cambiar desde un estado i a otro nuevo estado j en n pasos se define como:

$$p_{ij}^{(n)} = \Pr(X_n = j | X_0 = i) \tag{7.12}$$

7.1.6.3. Recurrencia

Un estado i es recurrente (persistente) cuando no existe una probabilidad distinta de cero de que, partiendo de i volvamos en el futuro a i. Por tanto un estado i será recurrente sí y solo sí:

$$\sum_{n=0}^{\infty} p_{ij}^{(n)} = \infty \tag{7.13}$$

Sin embargo y a pesar de que en un estado recurrente sea posible volver, debemos tener en cuenta que la esperanza de volver a dicho estado no tiene por que serlo. Sólo cuando la esperanza de volver a un estado sea finita llamaremos al estado *positivamente recurrente*.

Cuando un estado no sea recurrente, será *transciente* si existe una posibilidad distinta de cero de no poder volver a i partiendo de i; o *absorvente* cuando sea imposible abandonar ese estado (será absorvente sí y solo sí $p_{ii} = 1$ y $p_{ij} = 0 \, \forall i \neq j$).

7.1.6.4. Reducibilidad

Se considera que un estado j es accesible por otro estado distinto i (escrito $i \rightarrow j$) sí, estando en el estado i existe una probabilidad distinta de cero de alcanzar j:

$$\Pr(X_n = j | X_0 = i) > 0 \tag{7.14}$$

Dicha probabilidad no puede ser cero por que cada estado puede ser alcanzado desde sí mismo.

Por otro lado, se dice que i se comunica con j ($i \leftrightarrow j$) cuando j es accesible por i y viceversa. Sí además, todos los posibles pares de estados de un conjunto C se comunican entre sí, diremos que C es una *clase comunicada*. Cuando C es igual al espacio de estados de una cadena de Markov, dicha cadena será *irreducible* y por tanto en dicho caso será posible alcanzar cualquier estado desde cualquier otro estado.

7.1.6.5. Periodicidad

Un estado i tendrá un periodo k sí para volver a dicho estado se requieren k instantes. Se define la periodicidad de un estado por:

$$k = mcd\{n : \Pr(X_n = i | X_0 = i) > 0\} \tag{7.15}$$

En el caso de que $k = 1$, se diría que el estado es *aperiódico*. Una propiedad de las clases comunicadas es que todo sus estados han de tener el mismo periodo k.

7.1.6.6. Ergodicidad

Un estado i será ergódico si es aperiódico y positivamente recurrente. Si todos los estados de una cadena de Markov son ergódicos, dicha cadena será *ergódica*.

7.1.6.7. Regularidad

Una cadena de Markov es regular si existe un entero n tal que:

$$p_{ij}^{(n)} > 0 \, \forall i, j \tag{7.16}$$

En cualquier otro caso, la cadena será *irregular*.

7.1.7. Variantes

Existen, a parte de los modelos ocultos de Markov, otras variantes de los procesos de Markov.

7.1.7.1. Cadenas de Markov homogéneas en el tiempo

Cuando un sistema de Markov es homogéneo en el tiempo, la intensidad de cambiar a otro estado en un instante dado es constante en el tiempo; las probabilidades de transición dependen solo del estado en que nos encontremos, pero no del tiempo:

Sea $\{X(t), t \geq 0\}$ un proceso de Markov, si las probabilidades condicionadas $\Pr(X(s+t) = j | X(s) = i)$ para $s, t \geq 0$, no dependen de s entonces se dice que el proceso es homogéneo en el tiempo y sus probabilidades de transición en el tiempo vienen dadas por:

$$a_{ij}(t) = \Pr(X(t) = j | X(0) = i) \tag{7.17}$$

Siendo la matriz de transición $A(t)$ cuyo elemento (i, j) es $a_{ij}(t)$. Es importante tener en cuenta que entre las propiedades de esta matriz está que:

- $a_{ij}(0) = 1$
- $a_{ij}(0) = 0$ para $i \neq j$
- $a_{ij} \geq 0$
- $\sum a_{ij} = 1$

Como en muchos casos es necesario estudiar el tiempo entre ocurrencias en un sistema de Markov, podemos utilizar la siguiente definición: Sea$\{X(t), t \geq 0\}$ un proceso de Markov y los instantes $0 \leq T_1 < T_2 < T_3 < \ldots$ tales que en el instante T_i el proceso haga una transición entre dos estados, el tiempo entre ocurrencias (duración) será: $Y_n = T_n - T_{n-1}$, siendo $T_0 = 0$.

7.1.7.2. Cadenas de Markov de orden m

Se dice que una cadena de Markov es de orden m o tiene una memoria $m$cuando para todo n existe un m finito tal que:

$$\Pr(X_n = x_n | X_{n-1} = x_{n-1}, X_{n-2} = x_{n-2}, \ldots, X_1 = x_1) = \tag{7.18}$$

$$= \Pr(X_n = x_n | X_{n-1} = x_{n-1}, X_{n-2} = x_{n-2}, \ldots, X_1 = x_1) \tag{7.19}$$

En estas cadenas se puede transformar para extraer una nueva cadena que cumpla la propiedad habitual de Markov.

7.1.8. Cadenas de Markov con espacio de estados finito

Las cadenas de Markov con espacios finitos están definidas por los siguientes parámetros:

Matriz de Transición Si la cadena de Markov tiene un espacio de estados finito, sus probabilidades de transición se pueden representar en una matriz, la *matriz de transición*, cuyos elementos se definen por:

$$a_{ij} = \Pr(X_{n+1} = j | X_n = i) \tag{7.20}$$

Siendo a_{ij} las probabilidades de transición de todos los estados que dan lugar a la matriz de transición A; siendo esta una una matriz estocástica derecha y homogénea en el tiempo. Esta última propiedad implica que dicha matriz será la misma en cada transición, por tanto podrá calcularse la probabilidad de transición desde un estado a otro en k pasos como la k-ésima potencia de la matriz, A^k.

Vector de Probabilidades Iniciales Se da una distribución inicial en un vector fila, π, que no cambia con la aplicación de dicha matriz. Dicho vector es el vector propio de la matriz asociado con el valor propio 1:

$$\pi A = \pi \tag{7.21}$$

Dicho vector de probabilidad estacionaria existe (demostrable mediante el teorema de Perron-Frobenius) y su mayor autovalor en una matriz estocástica es siempre 1. Dicho vector existe pero no tiene por que ser único, salvo que la cadena de Markov sea irreducible y aperiódica, cuyo caso la distribución estacionaria será única, en este caso se puede calcular tomando el límite (puede ser algo complejo de calcular por que hay algunos casos especiales):

$$\lim_{k\to\infty} (A^k)_{ij} = \pi_j \tag{7.22}$$

ó

$$\lim_{x\to\infty} A^k = 1\pi \tag{7.23}$$

Siendo:

$$1\pi = \begin{pmatrix} \pi \\ \pi \\ \pi \\ \vdots \\ \pi \end{pmatrix} = Q \tag{7.24}$$

Es posible calcular Q cuando no es posible obtener $\lim_{x\to\infty} A^k = 1\pi$ mediante el siguiente desarrollo: Sea $QP = Q$, si restamos Q a ambos lados de la ecuación, tenemos:

$$Q(P - I_n) = 0_{n,n} \tag{7.25}$$

Como el resultado de multiplicar dos matrices estocásticas es también una matriz estocástica, a veces es posible calcular Q usando el siguiente desarrollo: podemos usar la función f(X) que devuelve la matriz X en la cual su columna derecha sea ha reemplazado por una columna de 1's:

$$X = \begin{pmatrix} a_{11} & a_{12} & \dots & a_{1j} & b \\ a_{21} & a_{22} & \dots & a_{2j} & c \\ & & \ddots & \vdots & d \\ a_{i1} & ai_2 & \dots & a_{ij} & e \\ \dots & \dots & \dots & \dots & f \end{pmatrix} \Longrightarrow f(X) = \begin{pmatrix} a_{11} & a_{12} & \dots & a_{1j} & 1 \\ a_{21} & a_{22} & \dots & a_{2j} & 1 \\ & & \ddots & \vdots & 1 \\ a_{i1} & ai_2 & \dots & a_{ij} & 1 \\ \dots & \dots & \dots & \dots & 1 \end{pmatrix} \tag{7.26}$$

Y así podemos usar la salida de dicha función para evaluar la siguiente ecuación:

$$Q = f(0_{n,n})[f(P - I_n)]^{-1} \tag{7.27}$$

También es posible que dicha inversa no exista y se tenga que recurrir a otros métodos.

7.1.9. Procesos de Markov

Las cadenas de Markov se incluyen dentro de los denominados procesos estocásticos, utilizados para estudiar el comportamiento de variables aleatorias a lo largo del tiempo. El interés de los procesos estocásticos es el de modelar el comportamiento de un sistema y su operación durante algunos periodos de tiempo, de esta manera se pueden procesar las señales generadas por algún proceso real para así analizar su comportamiento en el tiempo.

Dado que existen muchos tipos de señales (dependiendo de si sus propiedades varían o no en el tiempo, de sus fuentes, del ruido y otras características) es muy interesante contar con algún modelo que simplifique su análisis y clasificación.

La razón principal detrás del uso de modelos en el análisis de señales (así como en cualquier campo científico) es la de obtener una base que permita describir el comportamiento del sistema y así poder elaborar una cierta descripción teórica que nos permita describir que tipo de procesamiento es necesario aplicar a la señal para obtener una salida que nos sea de utilidad. Además, el uso de modelos nos permite obtener información acerca del comportamiento de la fuente de la señal aunque no podamos observar esta de forma directa, con lo que es posible utilizar un modelo para simular dicha fuente y aprender de las simulaciones.

Mientras que en el caso de señales determinísticas (que tienen determinadas propiedades conocidas) es fácil estimar los valores de salida a partir de sus parámetros, en el caso de señales estocásticas su análisis se complica dado que dichas señales son básicamente procesos aleatorios, de forma que es necesario aplicar algún tipo de procesado que nos permita clasificarlas.

Una de las formas de clasificar las señales de un sistema estocástico es utilizando cadenas de Markov, que son procesos estocásticos en el que los valores del tiempo son discretos y los estados posibles de la variable aleatoria contiene valores discretos, esto es, es una cadena estocástica de tiempo discreto.

Las cadenas de Markov, se clasifican, además, dentro de los procesos estocásticos de Markov, que son aquellos en el que el estado futuro de un proceso es independiente de los estados pasados y solamente depende del estado presente. De esta manera las probabilidades de transición entre los estados para los instantes t y $t+1$ solamente depende de los estados que la variable adquiere en dichos instantes.

7.2. Modelos Ocultos de Markov

En el punto anterior (7.1.5 en la página 51) se han explicado unos modelos de Markov en los cuales el estado actual es directamente observable, lo cual limita dichos modelos en muchos problemas. El modelo oculto de Markov (*HMM*) extiende el modelo de Markov de manera que el estado del modelo no es directamente observable, en vez de eso se puede observar una cierta salida que depende del estado en que se encuentre el modelo, de esta forma el modelo se convierte en sistema estocástico doble en el que existe un proceso estocástico oculto (de ahí el oculto) que solo puede ser observado indirectamente. De esta manera, las salidas generadas por un HMM ofrecen cierta información acerca de la secuencia de estados que ha ido siguiendo el modelo, pero no permiten observar sus estados directamente. Los elementos que caracterizan un HMM

son:

1. Número de estados del modelo, N: si bien los estados del modelo no son directamente observables, suele ser interesante que estos representen algún tipo de característica de la fuente que origine las señales observadas. Los posibles estados se denotan por: $S = \{S_1, S_2, \dots S_N\}$ y llamamos X_t al estado en que se encuentra el modelo en el instante t. Se representa por X_t por que no se puede saber exactamente a qué estado pertenece realmente por no ser observable directamente.

2. Número de observaciones distintas por cada estado, M, que corresponden a la salida física del sistema modelado. M representa el tamaño del alfabeto discreto del modelo: $V = \{v_1, v_2, \dots v_M\}$.

3. La distribución de probabilidades de transición,

$$A = \{a_{ij}\} \tag{7.28}$$

   Siendo:

$$a_{ij} = \Pr(X_{t+1} = S_j | X_t = S_i), \forall\, i \geq 1,\, j \leq N \tag{7.29}$$

   Existen HMM en que posible alcanzar cualquier estado a partir de otro estado previo ($a_{ij} > 0 \forall i, j$), mientras que en otras es posible que existan $a_{ij} = 0$, de forma que algunos estados no sean directamente alcanzables.

4. Distribución de probabilidades de observación. Se define como:

$$B = \{b_j(k)\} \tag{7.30}$$

   cuando para un estado j, se cumple:

$$b_j(k) = \Pr(v_k,\, en\, t | X_t = S_j), siendo 1 \leq j \leq N y 1 \leq k \leq M \tag{7.31}$$

5. Distribución inicial de estados:

$$\pi = \Pr(X_1 = S_1) \tag{7.32}$$

   cuando $1 \leq i \leq N$.

A partir de los valores anteriores de $N, M, A, B\, y\, \pi$ se utiliza el HMM bien como generador de una secuencia de observaciones o bien como modelo para analizar si dicha secuencia de observaciones pudo ser generada por dicho HMM. Siendo T el número total de observaciones de la secuencia, se define O como:

$$O = O_1, O_2, \dots O_T \tag{7.33}$$

Para obtener o clasificar dicha secuencia se siguen estos pasos:

1. Se elije un estado inicial $X_1 = S_i$ a partir de la distribución inicial, .

2. Se inicia $t = 1$.

3. Se elije $O_t = v_k$ atendiendo a la distribución de probabilidad B.

4. Se cambia a un nuevo estado $X_{t+1} = X_j$ atendiendo a la distribución de probabilidad A.

5. Se actualiza $t = t + 1$ si $t < T$ y se vuelve al punto 3, caso contrario se finaliza.

Así, para representar una HMM es necesario especificar los parámetros N y M, la secuencia de observaciones O (7.33) y las tres distribuciones de probabilidad (7.32, 7.28 y 7.30) que, para simplificar, representaremos con la notación:

$$\lambda = (\pi, A, B) \tag{7.34}$$

A modo de ejemplo utilizaremos un HMM que modela el comportamiento de una pareja de dados de seis caras, uno de ellos sigue una distribución normal, en que todas sus caras tienen la misma posibilidad de aparecer y el otro está trucado, este HMM tendría $N = 2$ estados, siendo el estado $S_1 = 1$ una tirada del dado normal y $S_2 = 2$ una tirada del dado trucado:

$$S = \{1, 2\} \tag{7.35}$$

Cada estado tiene $M = 6$ posibles observaciones, correspondientes a las salidas de los dados y tendría la siguiente matriz de transición:

$$A = \left\{ \begin{array}{cc} 0{,}95 & 0{,}05 \\ 0{,}4 & 0{,}6 \end{array} \right\} \tag{7.36}$$

Las posibilidades de observar cada una de las salidas del dado dependen del estado que se está utilizando, para el caso del dado normal, la matriz de posibilidades de observación es:

$$B_1 = \left[\begin{array}{cccccc} 1/6, & 1/6, & 1/6, & 1/6, & 1/6, & 1/6 \end{array} \right]$$

El segundo dado en cambio está trucado y presenta las siguientes posibilidades de observación:

$$B_2 = \left[\begin{array}{cccccc} 1/10, & 1/10, & 1/10, & 1/10, & 1/10, & 1/2 \end{array} \right]$$

Así, la matriz de posibilidades de observación de este HMM sería:

$$B = \left\{ \begin{array}{cccccc} 1/6 & 1/6 & 1/6 & 1/6 & 1/6 & 1/6 \\ 1/10 & 1/10 & 1/10 & 1/10 & 1/10 & 1/2 \end{array} \right\} \tag{7.37}$$

Finalmente, utilizamos unas probabilidades iniciales:

$$\pi = \left\{ \begin{array}{cc} 0{,}95 & 0{,}05 \end{array} \right\} \tag{7.38}$$

Así nuestro HMM estará caracterizado por:

$$\lambda_{hmm} = \left\{ \left\{ \begin{array}{cc} 0{,}95 & 0{,}05 \\ 0{,}1 & 0{,}9 \end{array} \right\} \left\{ \begin{array}{cccccc} 1/6 & 1/6 & 1/6 & 1/6 & 1/6 & 1/6 \\ 1/10 & 1/10 & 1/10 & 1/10 & 1/10 & 1/2 \end{array} \right\} \left\{ \begin{array}{cc} 0{,}95 & 0{,}05 \end{array} \right\} \right\} \tag{7.39}$$

La razón por la que se usará dicho ejemplo es que resulta fácil de entender y se pude explicar bien su comportamiento, por ejemplo se puede observar que funciona como la mezcla de dos distribuciones normales con medias ligeramente distintas: si bien esto lo haría candidato para resolverlo por otros medios, al estar mezclado de forma que se tienda a permanecer más tiempo en el estado correspondiente al dado normal y no al dado cargado y al estar sus medias bastante cercanas como se puede apreciar en los siguientes histogramas:

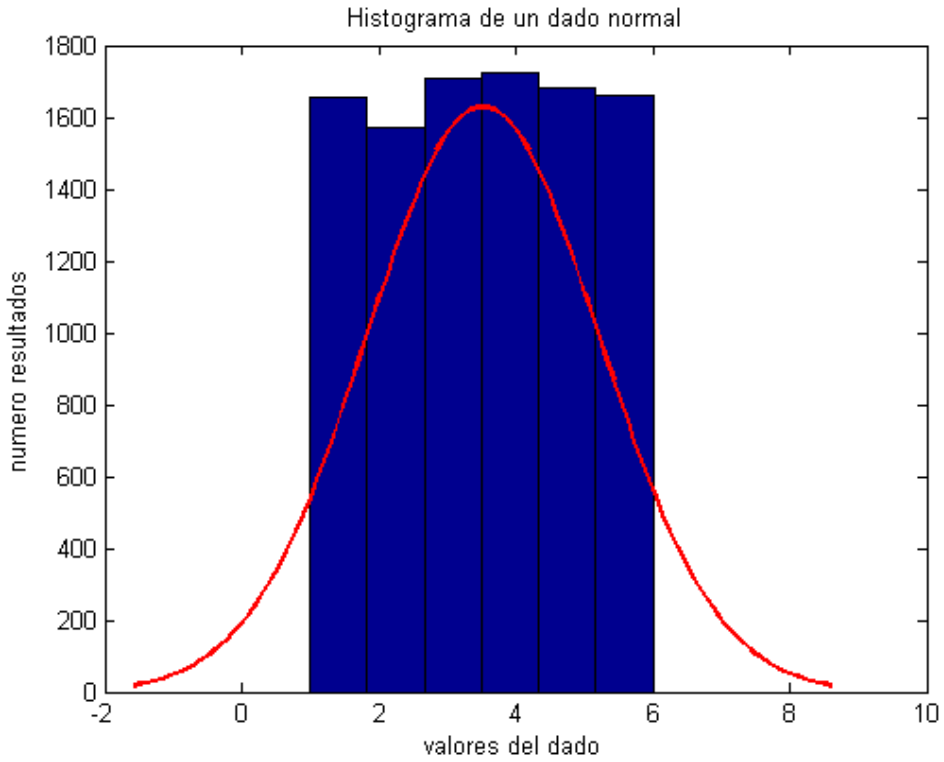


Figura 7.2: Histograma de un dado no cargado

Se puede ver que todos los valores tienen las mismas probabilidades, en cambio en el caso de un dado cargado tendríamos:

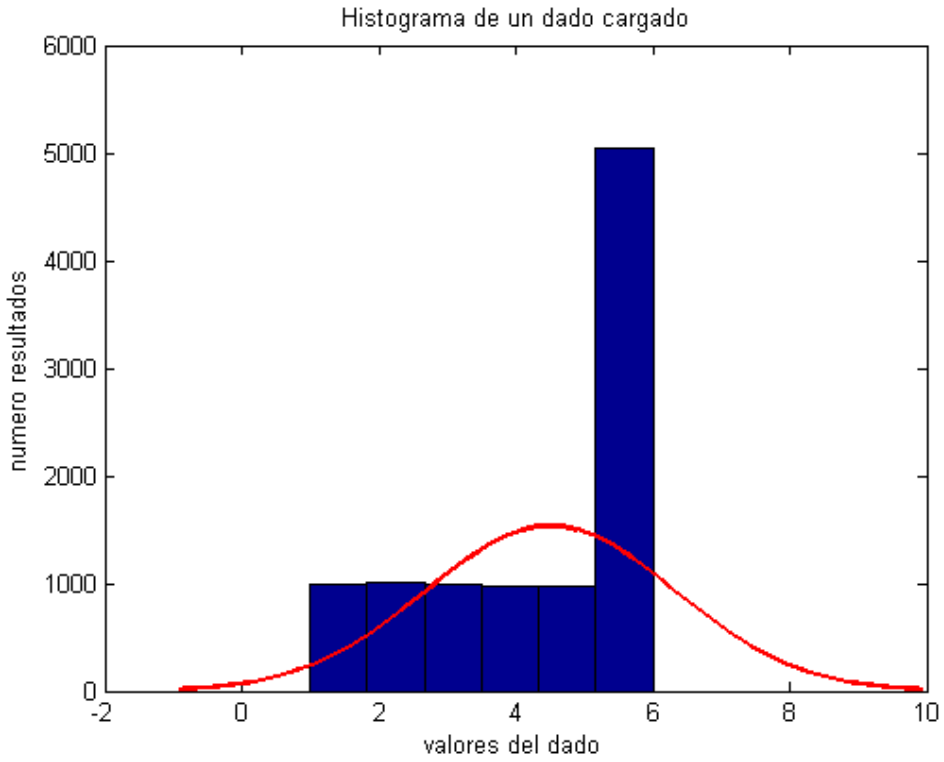


Figura 7.3: Histograma de un dado cargado

En un histograma de una secuencia larga (unas 10000 tiradas), la salida de nuestro HMM permitiría observar con claridad que no es una distribución normal, bastaría simplemente con comprobar la media, cercana a 4, como se puede observar en la siguiente figura (figura 7.4 en la página siguiente), en la cual el valor cargado del dado se repite muchas más veces que el resto.

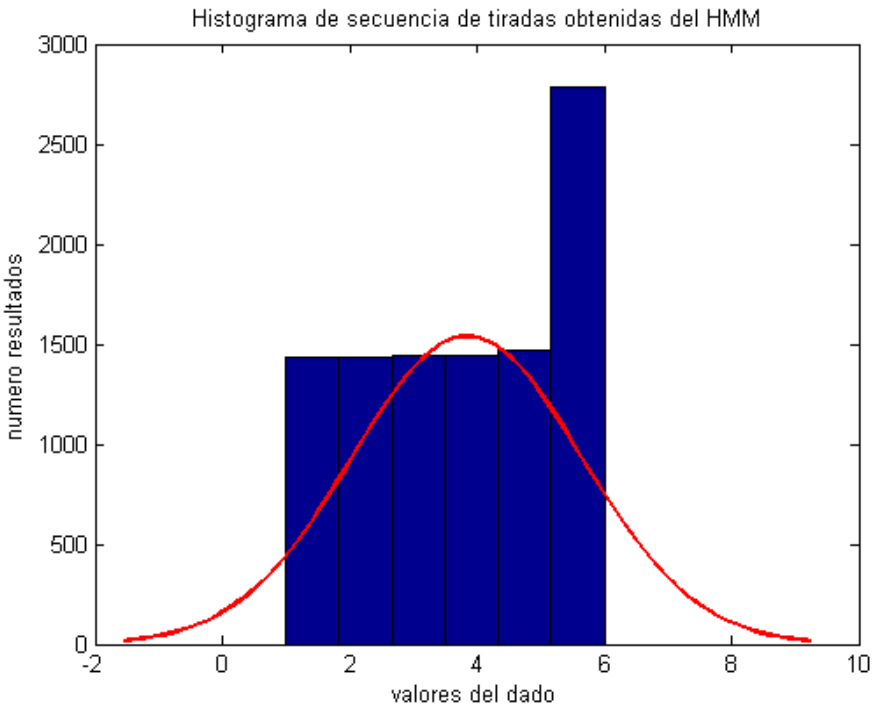


Figura 7.4: Histograma de una secuencia larga de tiradas mezclando un dado normal y otro cargado

Ahora bien, si nos enfrentáramos a un numero mucho menor de tiradas, utilizando simplemente la media sería difícil comprobar si estamos o no ante un dado cargado, así con una partida con 60 tiradas, obtenemos un histograma como el mostrado en 7.5 en la página siguiente, cuya media es de $3,72$; lo bastante cercana a 3 como para que parezca que se ha utilizado un dado no cargado:

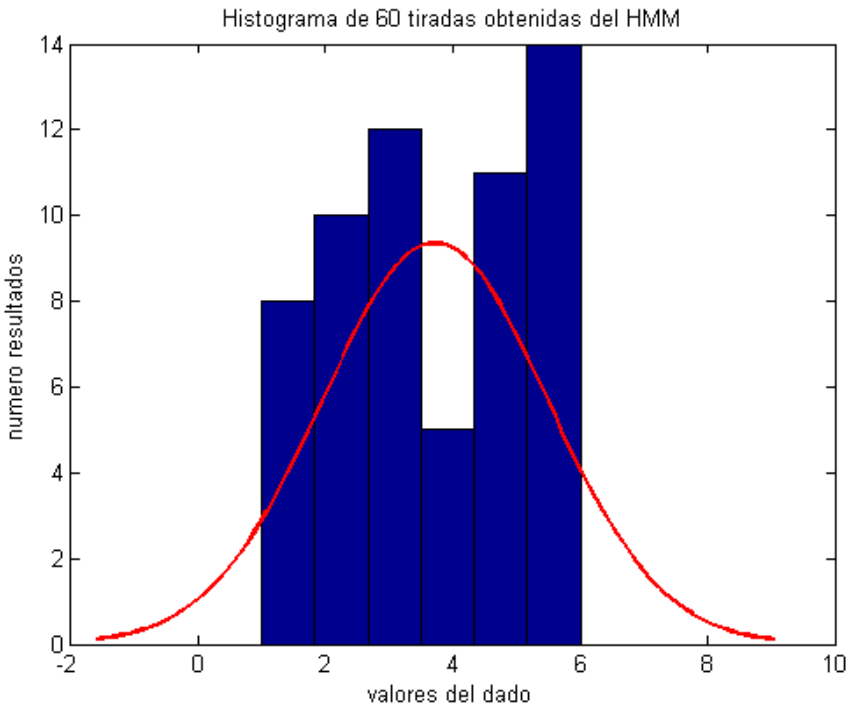


Figura 7.5: Histograma de 60 tiradas, mezclando un dado normal y otro cargado

En este último caso, al haber pocos valores resulta muy complicado analizar si realmente se ha utilizado o no dados cargados dada la variación entre los diversos valores, por lo que utilizaremos los algoritmos que vamos a definir a fin de comprobar si esta secuencia corresponde o no a un HMM y que secuencia de estados la habría generado.

Finalmente y a modo de ejemplo utilizaremos para cada uno de los siguientes ejemplos una determinada secuencia de observaciones, O (7.2), que definiremos en cada problema, a fin de resaltar las características de cada problema.

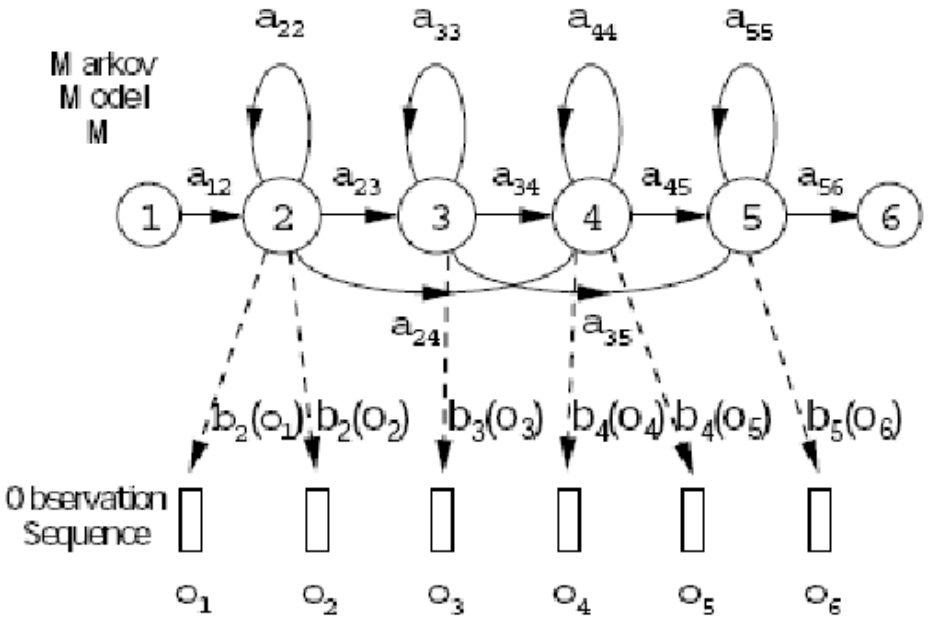


Figura 7.6: Autómata representando un modelo oculto de Markov

7.3. Probabilidad de que una secuencia de observaciones sea generada por el modelo

Dado un modelo y una secuencia de observaciones, este problema intenta calcular la probabilidad de que dicha secuencia haya sido generada por el modelo. Por ejemplo, la solución a este problema nos permitiría elegir entre varios modelos, el que mejor es capaz de generar dicha secuencia de observaciones. La probabilidad de observar una secuencia de observaciones L dada por $O = O_1, O_2, \ldots O_T$ usando el modelo, requeriría enumerar todas las posibles secuencias de estados de longitud L:

$$S = s_1 s_2 \ldots s_L \tag{7.40}$$

Siendo s_1 el estado inicial, así la probabilidad de observar dicha secuencia sería:

$$\Pr(O|O|S, \lambda) = \prod_{T=1}^{T} \Pr(O_t|s_t\lambda) \tag{7.41}$$

Tras sumar las probabilidades condicionadas sobre los posibles estados es:

$$\Pr(O) = \sum_{S} \Pr(O|S) \Pr(S) \tag{7.42}$$

Dado que dicho problema es de una complejidad intratable, por que calcularla requiere sumar todas las posibles secuencias de estados, es necesario resolver el problema de otra manera. La solución a este problema está en usar algoritmo forward-backward, que permite obtener un resultado eficientemente:

1. Definimos

$$\alpha_t(i) = \Pr(O_1 O_2 \dots O_t, X_t = S_i | \lambda) \tag{7.43}$$

   Siendo la probabilidad de la observación parcial de la secuencia hasta el instante t con un estado S_i en el instante t dado el modelo.

2. Iniciamos:

$$\alpha_1(i) = \pi b_i(O_i)\, 1 \leq i \leq N \tag{7.44}$$

   Se inicia las probabilidades de avanzar como una probabilidad conjunta entre la observación respecto al estado.

3. Inducción:

$$\alpha_{t+1}(j) = [\sum_{i=1}^{N} \alpha_t(i) a_{ij}] b_j(O_{t+1})\, 1 \leq t \leq T,\, 1 \leq j \leq N \tag{7.45}$$

   Con este paso se calcula la probabilidad $\alpha_t(i)$ de que se observe el evento $O_1 O_2 \dots O_t$.

   Considerando al estado S_i como el estado en que nos encontramos en el instante t, el producto de probabilidades $\alpha_t(i) a_{ij}$ será igual a la probabilidad de observar $O_1 O_2 \dots O_t$ y alcanzar el estado S_i en el instante $t+1$. La suma de todas estas probabilidades para los posibles estados $1 \leq j \leq N$ da como resultado la probabilidad de que se alcance el estado S_i en el instante $t+1$ habiendo realizado todas las observaciones de los estados previos y actual.

4. Finalización

$$\Pr(O|\lambda) = \sum_{i=1}^{N} \alpha_T(i) \tag{7.46}$$

   Una vez obtenidos todos los resultados de la suma anterior, se suman las probabilidades obteniendo como resultado la probabilidad de que dado un HMM representado por los siguientes parámetros (fórmulas 7.34, 7.28 en la página 57, 7.30 en la página 57 y 7.32 en la página 57),

$$\lambda = (A, B, \pi) \tag{7.47}$$

   para los valores observemos la secuencia O (fórmula 7.33 en la página 57). Con este proceso, hemos calculado la probabilidad de la secuencia (*forward*), ahora bien podemos aprovechar estos cálculos para calcular la probabilidad hacia atrás (*backward*) que se utilizará en el problema de la búsqueda de secuencias óptimas:

$$\beta_T(i) = \Pr(O_{t+1}O_{t+2}\dots O_T|X_t = S_i, \lambda) \tag{7.48}$$

Siendo esta la probabilidad de una observación parcial desde $t+1$ hasta el final de la secuencia dados un estado S_i en el instante t y el modelo λ cuyo cálculo se realiza utilizando las siguientes ecuaciones:

1. Inicializamos

$$\beta_T(i) = 1,\ 1 \leq i \leq N \tag{7.49}$$

2. Inducción:

$$\beta_T(i) = \sum_{j=1}^{N} a_{ij}b_j(O_{t+1})\beta_{t+1}(j),\ t = T-1, T-2\dots 1,\ 1 \leq i \leq N \tag{7.50}$$

   Calcula las posibilidades de observar la secuencia en $t+1$ teniendo en cuenta los posibles estados que puede tomar S_j en $t+1$, las transiciones desde S_i hasta S_j (esto es, a_{ij}) la posible observación O_{t+1} dado el estado j (esto es, $b_j(O_{t+1})$) y el resto de observaciones parciales dado el estado j (el termino representado por $\beta_{t+1}(j)$).

Así, queremos saber que probabilidades hay de que el HMM 7.2 en la página 59 haya generado dos secuencias de longitud 60, la primera de ellas es la que ha generado la salida descrita en 7.5 en la página 62 y la segunda es una secuencia de 60 tiradas con valor '1':

$$O_1 = [6,2,2,3,5,6,1,6,2,2,6,6,6,6,2,6,6,6,1,1,5,3,6,3,4,1\dots] \tag{7.51}$$

$$O_2 = [1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1\dots] \tag{7.52}$$

Si utilizamos la función *dhmm_ logprob* para obtener la probabilidad (logarítmica) de que la secuencia de observaciones O (fórmula 7.33 en la página 57) se haya obtenido por 7.2 en la página 59 con la siguiente llamada:

dhmm_ logprob(O, π, A, B)

Obtenemos los siguientes resultados:

dhmm_ logprob(O_1, π, A, B)=-104.8170

dhmm_ logprob(O_2, π, A, B)=-110.0784

Así, la secuencia O_1 es más probable que haya sido generada por 7.2 en la página 59que la secuencia O_2.

7.4. Problema de la decodificación (búsqueda de la secuencia óptima dada una observación)

Este problema consiste en intentar descubrir la parte oculta del modelo de forma que encontremos la secuencia correcta de estados que dan lugar a una salida en el modelo.

Desgraciadamente no existe algo que podamos llamar una secuencia "correcta" de estados que genere una salida dada, por lo que se intenta solucionar este problema de la mejor manera posible, ahora bien, es necesario imponer algunos criterios a la hora de intentar descubrir el estado de forma que los criterios escogidos harán variar el resultado final.

Una de las aproximaciones comúnmente utilizada consiste en buscar la secuencia de estados que han dado lugar a una observación, para conseguir esto definimos la probabilidad de estar en el estado S_i en instante t dada la secuencia de observaciones O y el modelo como:

$$\gamma_t(i) = \Pr(X_t = S_i|O, \lambda) \tag{7.53}$$

Que expresada como las variables forward-backward es:

$$\gamma_t(i) = \frac{\alpha_t(i)\beta_t(i)}{\Pr(O|\lambda)} = \frac{\alpha_t(i)\beta_t(i)}{\sum_{i=1}^{N} \alpha_t(i)\beta_t(i)} \tag{7.54}$$

Siendo $\alpha_t(i)$ la probabilidad de observar la secuencia $O_1O_2 \dots O_t$ en el estado S_i en el instante t mientras que $\beta_t(i))$ es la probabilidad hacia atrás (*backward*) explicada en el problema de la evaluación y que tiene en cuenta el resto de observaciones a realizar $O_{t+1}O_{t+2} \dots O_T$ dado el estado S_j en el instante t. El divisor $\Pr(O|\lambda)$ se utiliza como factor de normalización a fin de asegurar que el producto $\alpha_t(i)\beta_t(i)$ haga que los sucesivos $\gamma_t(i)$ cumplan que:

$$\sum_{i=1}^{N} \gamma_t(i) = 1 \tag{7.55}$$

Así, si utilizamos el valor $\gamma_t(i)$ obtenido, podemos calcular el estado X_t más probable en el instante t:

$$X_t = \text{máx}(\gamma_t(i))_{1 \leq i \leq N,\, 1 \leq t \leq T} \tag{7.56}$$

Debe notarse que aunque dicha formula maximice el número esperado de estados, es posible que el resultado tenga secuencias sin sentido debido a que determina el estado más probable en un instante dado pero sin tener en cuenta la probabilidad de ocurrencia de una secuencia dada: pueden llegar a darse transiciones con una probabilidad '0'.

Como se ha indicado anteriormente, esto se puede solucionar cambiando el criterio de optimización, aunque de forma genérica lo que se suele hacer es buscar el mejor camino posible tal que se maximice $\Pr(X|O, \lambda)$, equivalente también a maximizar $\Pr(X, O|\lambda)$, que puede resolverse eficientemente con el algoritmo de *Viterbi*:

Para buscar la mejor secuencia de estados $X = \{X_1X_2 \dots X_T\}$ que defina la observación $O = \{O_1O_2 \dots O_T\}$ debemos definir la función de cantidad δ como

la probabilidad más alta sobre un camino dado en tiempo t que tiene en cuenta las primeras t observaciones y finaliza en el estado S_i.

$$\delta_t(i) = \max_{X_1 X_2 \ldots X_{t-1}} \Pr(X_1 X_2 \ldots X_T = i,\, O_1 O_2 \ldots O_t | \lambda) \tag{7.57}$$

Y por inducción:

$$\delta_{t+1}(j) = [\max \delta_t(i) a_{ij}] \cdot b_j(O_{t+1}) \tag{7.58}$$

Para obtener la secuencia se usa un vector, ψ, que almacena los δ maximizados por cada variable t y j:

1. Inicializa

$$\delta_t(i) = \pi_i b_j(O_1)\, 1 \leq i \leq N \tag{7.59}$$

$$\psi_1 = 0 \tag{7.60}$$

2. Recursión

$$\delta_t(j) = \max_{1 \leq i \leq N} [\delta_{t-1}(i) a_{ij}] \cdot b_j(O_t) \;\; 2 \leq t \leq T,\, 1 \leq j \leq N \tag{7.61}$$

$$\psi_t(j) = \max_{1 \leq i \leq N} [\delta_{t-1}(i)] \cdot a_{ij} \;\; 2 \leq t \leq T,\, 1 \leq i \leq N \tag{7.62}$$

3. Finalización

$$p^* = \max_{1 \leq i \leq N} [\delta_t(i)] \tag{7.63}$$

$$X_t^* = \arg \max_{1 \leq i \leq N} [\delta_t(i)] \tag{7.64}$$

4. Obtención de la secuencia por backtracking

$$X_t^* = \psi_{t+1}(X_{t+1}^*) \;\; t = T-1, T-2 \ldots 1 \tag{7.65}$$

Ejemplo: Dado el HMM 7.2 queremos saber que combinación de tiradas de dados es la que más probablemente ha generado la siguiente secuencia:

$$O = \{6\,2\,2\,3\,5\,6\,1\,6\,2\,2\,6\,6\,6\,6\,2\,6\,6\,6\,1\,1\,5\,3\} \tag{7.66}$$

Si utilizamos la función de Matlab *viterbi*, descrita en el epígrafe 10.2.1 en la página 104 con la siguiente llamada:

viterbi(π, A, B, O)

Nos dará como resultado la siguiente secuencia:

$$X_t^* = \{1\,1\,1\,1\,1\,2\,2\,2\,2\,2\,2\,2\,2\,2\,2\,2\,2\,2\,1\} \tag{7.67}$$

Donde los 1 indican un dado normal y los 2 indican un dado cargado. Así, el HMM está clasificando la secuencia como compuesta por las tiradas de dos dados distintos, cada uno con una esperanza matemática distinta (en el dado normal todos los números tienen las mismas posibilidades de salir y en el cargado, el valor '6' tiene muchas más posibilidades de aparecer en cada tirada).

7.5. Entrenamiento

En este problema se intentan optimizar los parámetros del modelo de forma que pueda describir qué secuencia de estados genera una secuencia de observaciones. Se utiliza una secuencia de entrenamiento para ajustar el modelo de forma que pueda adaptar el modelo a una secuencia real. Desgraciadamente, no existe ningún algoritmo que permita resolver el problema de forma exacta en un tiempo tratable, pero existen algoritmos que permiten obtener un máximo local de probabilidad, aunque no siempre son capaces de encontrar el máximo global. Estos algoritmos son por tanto muy dependientes del punto en que se inician.

Los más conocidos son el algoritmo de Baum-Welch [4](caso especial de EM) o el Baldi-Chauvin [8], aunque nos centraremos en el primero por ser el más estudiado.

7.5.1. EM

El algoritmo EM busca máximos de de probabilidad para estimar los parámetros de un HMM. Se trata de un algoritmo iterativo que alterna dos pasos: un paso (E) donde se calcula la verosimilitud esperada tratando a las variables latentes como si fueran observables, y después, un paso de maximización (M), calcula los estimadores de máxima verosimilitud de forma que dichos parámetros maximicen la probabilidad hallada en el paso E. Los parámetros hallados en el paso M se usan en la siguiente iteración en el paso E. El algoritmo EM trata de buscar el MLE aplicando los siguiente pasos:

- Esperanza (E): calcula la esperanza de la verosimilitud respecto a la distribución condicional de z dados x y la actual estimación de parámetros $\theta^{(t)}$:

$$Q(\theta|\theta^{(t)}) = E_{z|x,\theta^{(t)}}[\log L(\theta, x, z)] \tag{7.68}$$

- Maximización (M): busca los parámetros que maximizan:

$$\theta^{(t+1)} = \arg_\theta \max Q(\theta|\theta^t) \tag{7.69}$$

7.5.2. Baum-Welch

Dada una secuencia de observaciones $O = O_1 O_2 \dots O_T$ el algoritmo Baum-Welch permite estimar los parámetros (λ) de un HMM que maximizan la probabilidad de dicha secuencia, $Pr(O_j)$. Sea $\xi_t(i,j)$ la probabilidad de estar en un estado S_i en el instante t y de cambiar al estado S_j en el instante $t+1$ dados un modelo y secuencia de observaciones O:

$$\xi_t(i,j) = \Pr(X_t = S_i, X_{t+1} = S_j|O, \lambda) \tag{7.70}$$

Si lo expresamos como variables *forward-backward*, tenemos que:

$$\xi_t(i,j) = \frac{\alpha_t(i)a_{ij}b_j(O_{t+1})\beta_{t+1}(j)}{\Pr(O|\lambda)} = \frac{\alpha_t(i)a_{ij}b_j(O_{t+1})\beta_{t+1}(j)}{\sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_t(i)a_{ij}b_j(O_{t+1})\beta_{t+1}(j)} \tag{7.71}$$

Si definimos $\gamma_t(i)$ como la probabilidad de estar en el estado S_i en el instante t dadas la secuencia de observación, O, y el modelo, λ, obtenemos que:

$$\gamma_t(i) = \sum_{j=1}^{N} \xi_t(i,j) \tag{7.72}$$

Si sumamos $\gamma_t(i)$ en cada instante de tiempo del intervalo $t = 1; 2 \ldots T-1$, obtenemos el número esperado de transiciones realizadas desde el el estado S_i:

$$\sum_{t=1}^{T-1} \xi_t(i,j) \tag{7.73}$$

Que es el número de transiciones esperado desde S_i hasta S_j. Usando dichas propiedades, es posible calcular una estimación de los parámetros de un HMM:

$$\pi_i \tag{7.74}$$

Será la frecuencia esperada en estado S_i en el instante $(t = 1) = \gamma_1(i)$ y

$$a_{ij} = \frac{\sum_{t=1}^{T-1} \xi_t(i,j)}{\sum_{t=1}^{T-1} \gamma_t(i)} \tag{7.75}$$

Es el numero de transiciones esperado desde S_i hasta S_j dividido entre el número de transiciones esperado desde S_i, y

$$b_j(k) = \frac{\sum_{t=1:O_t=O_k}^{T} \gamma_t(j)}{\sum_{t=1}^{T} \gamma_t(j)} \tag{7.76}$$

que es el número de veces que el modelo está en el el estado j y observando el evento O_k dividido entre el número de veces que se espera estar en el el estado j.

Utilizando este algoritmo es posible calcular un modelo, λ, estimado en cada iteración y repetir los cálculos con el nuevo modelo para mejorar la probabilidad de observar la secuencia de observaciones, O, con dicho modelo. El resultado final corresponde a la máxima probabilidad estimada (MLE) del HMM. También es posible calcular la reestimación maximizando la función auxiliar de Baum mediante técnicas de optimización, de forma que:

$$Q(\lambda, \lambda') = \sum_{Q} \Pr(Q|O,\lambda) \log[\Pr(O,Q|\lambda')] \tag{7.77}$$

De esta forma, cuando se maximiza $Q(\lambda, \lambda')$ se tiende a la máxima probabilidad estimada (MLE):

$$\max_{\lambda'}[Q(\lambda,\lambda') \Rightarrow \Pr(O|\lambda') \geq \Pr(O|\lambda)] \tag{7.78}$$

En cualquier caso, la reestimación es un caso especial del algoritmo EM en el cual los pasos a realizar son:

- Paso E (expectación): cálculo de la función Q(0;)
- Paso M (maximización): maximización sobre 0.

El inconveniente de utilizar este algoritmo en cualquiera de sus variantes es que el algoritmo de avance-retroceso sólo encuentra los máximos locales y en la mayor parte de problemas que se desean resolver utilizando HMM existen muchos de estos máximos locales.

7.5.3. Entrenamiento con múltiples secuencias de observaciones

En determinados tipos de HMM y en especial los modelos izquierda-derecha, no es posible entrenar el modelo utilizando una sola secuencia de observación y por tanto es necesario utilizar múltiples secuencias, cada una de las cuales con su propio factor de escalado. Esto se debe a que dichos modelos solo permiten un determinado número de observaciones por cada estado posible antes de saltar al siguiente estado, de manera que algunos estados pueden quedar vacíos cuando se utiliza una sola secuencia.

7.6. Optimización de Modelos Ocultos de Markov

Existen una serie de problemas prácticos a la hora de implementar un HMM como son los criterios de optimización, escalado, múltiples observaciones y otros que tienen que ver con la implementación a nivel práctico.

Los HMM tratan de modelar una señal (la secuencia de observación) en base a unos parámetros bien elegidos, sin embargo en aplicaciones reales esta aproximación puede ser incorrecta por varias razones: bien la señal tiene ruido o no es adecuada para las restricciones del HMM, o bien por que resulta muy complicado estimar de forma correcta todos los parámetros del HMM. Para este último problema existen varias alternativas al uso de la máxima probabilidad estimada (MLE) usada para estimar los parámetros. Existen además otras formas de entrenamiento que primero realizan búsquedas a mayor nivel, mientras que se realiza la búsqueda local al algoritmo de Baum-Welch, dichas técnicas se discutirán más adelante, en este punto se explicarán los posibles criterios de optimización.

7.6.1. Uso de múltiples HMM para dividir el problema

Esta alternativa se basa en la aproximación "divide y vencerás" y básicamente usa varios HMM entrenados a la vez de forma que las capacidades de discriminación del conjunto de todas ellas sean máximas. De esta manera el modelo resultante puede ser capaz de distinguir entre observaciones que han sido generadas por el modelo correcto, de aquellas que han sido generadas por otros modelos alternativos.

Con este enfoque, se obtienen V modelos $\lambda_v \ : \ v = 1; 2; \dots V$, cada uno capaz de distinguir determinadas secuencias de observación. Para lograr esto, se usan varias secuencias de observaciones $O_v \ : \ v = 1; 2; \dots V$para obtener los parámetros de los modelos λ_v de acuerdo al criterio de maximización:

$$\overset{*}{\underset{v}{\Pr}} = \max_{\lambda}(O^v|\lambda_v) \tag{7.79}$$

Si además se usa el criterio de maximizar la información mutua (MMI) entre los diferentes HMM de forma que la media de la información mutua (I) entre las distintas secuencias de observación O^v y todos los modelos $\lambda = (\lambda_1, \lambda_2 \dots \lambda_V)$ sea máxima. Esto se logra si:

$$I_v^* = \max_{\lambda}\{\log[\Pr(O^v|\lambda_v)] - \log[\sum_{w=1}^{V} \Pr(O^v|\lambda_w)]\} \tag{7.80}$$

De esta manera cada λ_v estaría lo bastante separada del resto de modelos de forma que sería capaz de reconocer algunas secuencias del conjunto O_v. Aunque en principio también sería posible obtener este resultado sumando todas la secuencias de entrenamiento de forma que:

$$I_v^* = \max_{\lambda}\{\sum_{v=1}^{V}[\log[\Pr(O^v|\lambda_v)] - \log[\sum_{w=1}^{V} \Pr(O^v|\lambda_w)]\} \tag{7.81}$$

Sin embargo, esta solución solo se puede obtener usando procedimientos de estimación generales, dado que intentar alcanzarla multiplica la complejidad del entrenamiento hasta hacerlo inviable en muchos casos, salvo los más simples.

7.6.2. Asumir que las señales analizadas pueden proceder de un proceso no Markoviano

Otra aproximación es partir de la base de que la señal observada pudo ser generada por un proceso de Markov pero en cambio obedece ciertas restricciones que permiten modelarla con un HMM. Así, es posible elegir los parámetros del HMM que minimicen la discriminación de información (DI), definida como la entropía entre las densidades de probabilidad de las señales válidas (conjunto Q) y que por tanto que satisfacen el HMM, y el conjunto de densidades de probabilidad de los HMM (conjunto P_v).

$$D(Q|P_v) = \int q(y) \ln[\frac{q(y)}{p(y)}]dy \tag{7.82}$$

Siendo p la función de densidad de probabilidad de P_v y q la función de densidad de probabilidad de Q. Existen técnicas que permiten calcular dicha solución (minimizar DI, MDI) para valores óptimos de λ. Es importante resaltar que todas las posibles aproximaciones de optimización, sea MEL, MMI o MDI se pueden expresar como aproximaciones MDI, si bien se diferencian en que cada una de las aproximaciones atribuyen una densidad de probabilidad distinta a la señal modelada o al modelo.

7.7. Escalado

El problema de escalado se debe a que cada uno de los términos t(i) consiste en una suma de términos a y b con valores menores que 1 y generalmente significativamente menores que 1 como la siguiente:

$$\prod_{S=1}^{t-1} a_{q_s q_s+1} \prod_{S=1}^{t-1} b_{q_s(O_s)} \tag{7.83}$$

De manera que para valores de $t > 10$, $\alpha_t(i)$ tiende a 0 exponencialmente y con $t > 100$ los valores son tan pequeños que resulta imposible calcularlos incluso usando precisión doble [1], por esta razón se hace necesario incluir un factor de escalado.

El procedimiento habitual es el de multiplicar tanto $\alpha_t(i)$ como $\beta_t(i)$ por un coeficiente independiente de i de forma que las escalas de $\alpha_t(i)$ y de $\beta_t(i)$ se mantengan en un ratio $1 \leq t \leq T$. Por ejemplo si utilizamos el coeficiente:

$$c_t = \frac{1}{\sum_{i=1}^{N} \alpha_t(i)} \tag{7.84}$$

Y calculamos con un t fijo,

$$\alpha_t(i) = \sum_{j=1}^{N} \alpha'_{t-1}(j) a_{ij} b_j(O_t) \tag{7.85}$$

Siendo el coeficiente escalado:

$$\alpha'_t(i) = \frac{\sum_{j=1}^{N} \alpha'_{t-1}(j) a_{ij} b_j(O_t)}{\sum_{i=1}^{N} \sum_{j=1}^{N} \alpha'_{t-1}(j) a_{ij} b_j(O_t)} \tag{7.86}$$

$$\alpha'_{t-1}(j) = (\prod_{\tau=1}^{t-1} c_\tau) \alpha_{t-1}(j) \tag{7.87}$$

Simplificando ambas ecuaciones, obtenemos:

$$\alpha'_t(i) = \frac{\alpha_t(i)}{\sum_{i=1}^{N} \alpha_t(i)} \tag{7.88}$$

$$\beta'_t(i) = c_t \beta_t(i) \tag{7.89}$$

Y una matriz de coeficientes tal que:

$$a'_{ij} \frac{\sum_{t=1}^{T-1} C_t \alpha_t(i) a_{ij} b_j(O_{t+1}) D_{t+1} \beta_{t+1}(j)}{\sum_{t=1}^{T-1} \sum_{j=1}^{N} C_t \alpha_t(i) a_{ij} b_j(O_{t+1}) D_{t+1} \beta_{t+1}(j)} \tag{7.90}$$

De manera similar se puede calcular π y B, siendo el único problema $\Pr(O|\lambda)$ dado que seguiría quedando fuera del rango de cálculo, en cualquier caso sabiendo que:

$$\prod_{t=1}^{T} c_t \sum_{i=1}^{N} \alpha_T(i) = C_T \sum_{i=1}^{N} \alpha_T(i) = 1 \tag{7.91}$$

Y por tanto,

$$\prod_{t=1}^{T} c_t \Pr(O|\lambda) = 1 \tag{7.92}$$

$$\Pr(O|\lambda) = \frac{1}{\prod_{t=1}^{T} c_t} \tag{7.93}$$

Con lo que se puede calcular:

$$\log[\Pr(O|\lambda)] = -\sum_{t=1}^{T} \log(c_t) \tag{7.94}$$

Como además, si se utiliza Viterbi para obtener la máxima probabilidad de una secuencia de estados, no hace falta escalar si utilizamos logaritmos:

$$\delta_t(i) = \max_{X_1, X_2 \ldots X_T} \{\log(\Pr X_1 X_2 \ldots X_T = i,\, O_1 O_2 \ldots O_t|\lambda)\} \tag{7.95}$$

$$\delta_1(i) = \log(\pi_i) + \log(b_i O_1) \tag{7.96}$$

$$\delta_t(i) = \max_{1 \leq i \leq N} \{\delta_{t-1}(i) + \log(a_{ij})\} + \log(b_j O_t) \tag{7.97}$$

$$\log P^* = \max_{1 \leq i \leq N} \delta_T(i) \tag{7.98}$$

La ventaja de esta aproximación, a parte de resolver el problema de los rangos de cálculo, es que requiere menos cálculos y que es posible realizar un precalculo de los términos $\log(a_{ij})$ de forma que no cuesten nada durante el cálculo de Viterbi. Además es posible precalcular $\log(b_j O_t)$ cuando se tiene una secuencia finita de observaciones, facilitando el cálculo de los resultados y reduciendo la complejidad del problema.

7.8. Modelos ocultos de Markov de Variable continua

Aunque hasta este punto hemos hablado solo de modelos ocultos de Markov que funcionan con entradas discretas a fin de facilitar su comprensión, sin embargo lo que nos interesa para nuestro clasificador son los modelos ocultos de Markov con entradas continuas, a fin de que puedan aceptar directamente las entradas de los residuos explicadas en 6 en la página 41.

La única diferencia entre una u otro tipo de entrada es la representación de las probabilidades de las variables ocultas: en el caso continuo, se utilizan funciones de densidad de probabilidad (gaussianas en nuestro caso, aunque pudiera ser que en otros problemas fuera necesario elegir otras funciones de probabilidad) en lugar de las distribuciones de probabilidades de observación definidas en 7.31 en la página 57, definimos:

$$b_j(\alpha_t) = \sum_{m=1}^{M} c_{jm} \mathrm{N}(\mu_{jm}, \sigma_{jm}, \alpha_t) \tag{7.99}$$

donde, para la suma de M gaussianas y con N estados (no confundir con la distribución normal en la anterior fórmula) utilizamos los siguientes parámetros:

- c_{jm} son los coeficientes de ponderación, siendo $c_{jm} \geq 0$, $1 \leq j \leq N$, $1 \leq m \leq M$ y $\sum_{m=1}^{M} c_{jm} = 1$, $1 \leq j \leq N$.

  En nuestros ejemplos utilizaremos el valor $c = I$ en todas nuestras pruebas (valor por defecto si no se especifica), de forma que las gaussianas a

utilizar tengan todas los mismos pesos; en cualquier caso, la herramienta desarrollada (ver) permitirá utilizar distintos valores.

- μ_{jm} es el vector con las medias de las gaussianas.
- σ_{jm} es la matriz de covarianza.

Así, cuando se trabaje con HMM continuos, el modelo se definirá como:

$$\lambda = (\pi,\, A,\, \mu_{jm},\, \sigma_{jm},\, c_{jm}) \tag{7.100}$$

Parte III

Metodologías de Evaluación

Capítulo 8

Evaluación de modelos

8.1. Problemas con la evaluación de modelos

Habitualmente, los modelos matemáticos son iterativos: se llega a la solución final tras una serie de pasos o iteraciones, cada cuál más cercana a una solución óptima. Sin embargo, no siempre es posible alcanzar una convergencia a dicho nivel óptimo por dos razones:

- El modelo converge al nivel óptimo solo en teoría. La convergencia teórica señala que hay un límite superior finito, pero sin indicar cómo de alto puede ser ese límite: debido a esto se puede llegar a un punto en que por mucho proceso que se realice, no se alcance la respuesta óptima.
- El modelo matemático es de tal complejidad que no se puede plasmar correctamente en un algoritmo: el modelo no es factible en términos de cálculo.

En nuestro caso, el modelo probabilístico de trayectorias (epígrafe 10 en la página 97) tiene, como ya veremos en la parte V en la página 109, problemas en ambos puntos, al igual que el algoritmo utilizado como grupo de control. En ambos casos, como en muchos otros casos en la vida real, existen toda una serie de problemas que no se resuelven con los evaluadores más "clásicos" o lo hacen de forma incompleta.

De forma general, una vez identificado un problema y una representación del mismo, si se desea aplicar un algoritmo capaz de decodificar (reconocer) una señal generada por el fenómeno modelado o si se desea generar un algoritmo que sea capaz de clasificar dichas señales es necesario evaluar el aprendizaje de dicho modelo, tanto para comprobar si está funcionando de forma correcta, como para ayudar en su posterior optimización (entrenamiento en 7.5.1 en la página 68).

Una vez obtenido un modelo que se considere válido de un problema es necesario comprobar la validez del mismo y también compararlo para ver como se comporta frente a otros modelos alternativos.

Para estas evaluaciones, necesitamos conocer:

- Tasas de error esperadas de un algoritmo de clasificación.
- Comparar dichas tasas frente a las tasas de error de otros algoritmos.

- Conocer cuándo son significativas las diferencias en precisión de un algoritmo.
- Finalmente, hemos de saber que si el número de datos no es significativo o es demasiado pequeño, obtendremos sesgos en la estimación que harán que las clasificaciones obtenidas no sean aplicables a futuras instancias.

Antes de explicar las motivaciones que nos han llevado a buscar una metodología de evaluación, explicaremos cuales son las herramientas más "clásicas" para la evaluación de modelos.

8.1.1. Estimación de la precisión de una hipótesis dada

Cuando se quiere evaluar la precisión de la hipótesis y conocer el error probable en la estimación de la precisión de dicha hipótesis se suele hacer a partir del cálculo de los errores de muestra y verdadero. Si partimos de un problema de aprendizaje en que dada una muestra de datos, con n ejemplos tomados aleatoriamente, siguiendo la distribución de probabilidad D, queremos saber si dada una hipótesis h, cuál sería la mejor estimación de la precisión de h sobre instancias futuras tomadas con la misma distribución y cuál sería el error probable en esta estimación de la precisión.

Así, tenemos dos posibles tipos de error:

- La tasa de error de la hipótesis sobre la muestra disponible, que es la que podemos calcular y
- La tasa de error de la hipótesis sobre toda la distribución desconocida D de ejemplos, que es la que quisiéramos calcular.

El error de muestra para la hipótesis h con respecto a la función f se define como:

$$error_s(h) = \frac{1}{n} \sum_{x \epsilon s} \delta(f(x), h(x)) \tag{8.1}$$

Siendo n el número de datos de la muestra y

$$\delta(f(x), h(x)) = \begin{cases} 1 & si\ f(x) = h(x) \\ 0 & si\ f(x) \neq h(x) \end{cases} \tag{8.2}$$

el error verdadero de una hipótesis es la probabilidad de que la hipótesis se equivoque para una instancia tomada aleatoriamente con la distribución D y se define como:

$$error_D(h) \equiv \Pr_{x \epsilon D}[f(x) \neq h(x)] \tag{8.3}$$

La probabilidad $\Pr_{x \epsilon D}$ (8.3) se toma siempre sobre una instancia de la distribución D.

Aunque lo que realmente se desea es conocer es el error verdadero $error_{D(h)}$ de la hipótesis, esto es imposible o impráctico, en cambio si que podemos medir el error de muestra $error_s(h)$ a partir de una muestra de los datos disponible. Así, deseamos saber cómo es de bueno el estimador $error_s(h)$ del error verdadero $error_D(h)$ para lo cuál usamos los intervalos de confianza.

8.1.2. Intervalos de confianza

Los intervalos de confianza son pares de números entre los cuales se estima que estará cierto valor desconocido con una determinada probabilidad. Dichos pares de números determinan un intervalo, calculado a partir de los datos de una muestra y el valor desconocido es un parámetro estadístico representativo de la población que se esté analizando.

La probabilidad de éxito en la estimación se representa por $1 - \alpha$ y se denomina nivel de confianza, siendo α la medida de las probabilidades de fallar en la estimación ante el intervalo dado (también denominado intervalo de significación). La variable es el llamado error aleatorio y es una medida de las posibilidades de fallar en la estimación mediante tal intervalo.

Como los niveles de confianza y las amplitudes de los intervalos varía de forma conjunta, intervalos más amplios tendrán más posibilidades de acierto (mayor nivel de confianza), mientras que los intervalos más pequeños, aumentan sus posibilidades de error, pero también ofrecen estimaciones más precisas.

Independientemente de la precisión buscada, es necesario conocer la distribución teórica que sigue el parámetro a estimar para poder calcular un intervalo de confianza. Habitualmente, dicho parámetro sigue una distribución normal aunque en otros casos puede ser necesario utilizar otro tipo de distribuciones, como la de Poisson (para medir números de sucesos ocurridos en intervalos de tiempo); en otros casos, es posible construir los intervalos a partir de la desigualdad de Chebyshov que establece que para una variable aleatoria X de media μ y varianza σ^2 la probabilidad de que X esté a un distancia $k > 0 \epsilon \mathbb{R}$ de su esperanza matemática es:

$$\Pr(|X - \mu| > k\sigma) \leq \frac{1}{k^2} \tag{8.4}$$

Sea cual sea la distribución de probabilidad del parámetro, el intervalo de confianza $[\vartheta_1, \vartheta]$ nos dará una estimación de dicho parámetro, ϑ, al $(1 - \alpha)\,\%$ si

$$P[\vartheta_1 \leq \vartheta \leq \vartheta_2] = 1 - \alpha \tag{8.5}$$

siendo P la función de distribución de probabilidad de ϑ.

8.1.3. Diferencias de error entre múltiples hipótesis

Si deseamos estimar una diferencia, d, entre los errores verdaderos de dos hipótesis, h_1 y h_2 para una función objetivo con valores discretos y dos conjuntos de prueba, s_1 y s_2 con los que probar dichas hipótesis, esto es:

- s_1, que con n_1 datos aleatorios prueba la hipótesis h_1, y
- s_2, con n_2 datos aleatorios prueba h_2

Podemos calcular d utilizando el procedimiento general para obtener intervalos de confianza:

$$d \equiv error_D(h_1) - error_D(h_2) \tag{8.6}$$

Como no podemos obtener los errores verdaderos como se ha explicado en 8.1.1 en la página anterior, hemos de usar los errores relativos de ambas hipótesis:

1. Calculamos el estimador no sesgado de d,

$$\hat{d} \equiv error_S(h_1) - error_S(h_2) \tag{8.7}$$

2. Si el tamaño de las muestras es lo bastante grande, $n_1, n_2 \geq 30$, los errores relativos $error_S(h_1)$ y $error_S(h_2)$ siguen distribuciones similares a la Normal.

3. Como la diferencia de 2 distribuciones de tipo Normal da como resultado una distribución Normal, entonces $\hat{d}$ seguirá una Normal con media d.

4. La varianza de la distribución será igual a la suma de las varianzas de la distribuciones $error_S(h_1)$ y $error_S(h_2)$, esto es:

$$\sigma_{\hat{d}}^2 \approx \frac{error_{S_1}(h_1)(1 - error_{S_1}(h_1))}{n_1} + \frac{error_{S_2}(h_2)(1 - error_{S_2}(h_2))}{n_2} \tag{8.8}$$

5. Para una variable aleatoria $\hat{d}$, con media $\mu_{\hat{d}}$ y varianza $\sigma_{\hat{d}}^2$ el estimado del intervalo de confianza de N para d es:

$$\hat{d} \pm z_N \sigma \tag{8.9}$$

$$\hat{d} \pm z_N \sqrt{\frac{error_{S_1}(h_1)(1 - error_{S_1}(h_1))}{n_1} + \frac{error_{S_2}(h_2)(1 - error_{S_2}(h_2))}{n_2}} \tag{8.10}$$

6. En caso de usar la misma muestra S para probar h_1 y h_2, entonces podremos calcular $\hat{d}$ como:

$$\hat{d} \equiv error_S(h_1) - error_S(h_2) \tag{8.11}$$

8.1.4. Pruebas de hipótesis

Si queremos probar que una hipótesis específica es cierta frente a otra dada, es posible hacerlo calculando la probabilidad de

$$\Pr(h_1, h_2) = error_D(h_1) - error_D(h_2) \tag{8.12}$$

dada una diferencia en errores de muestra $\hat{d} > 0$. Así, es posible calcular un $\hat{d}$ que caiga en un intervalo (para un solo lado de la media) de:

$$\hat{d} < \mu_{\hat{d}} + z_N \sigma_{\hat{d}} \tag{8.13}$$

Por ejemplo si queremos calcular la probabilidad de que $\hat{d}$ no sobreestime d con una probabilidad mayor a $0,1$,

$$\hat{d} < \mu_{\hat{d}} + 0,1 \tag{8.14}$$

expresamos dicho intervalo en función del número de desviaciones estándar permitidas, el valor que asegura que quedaría a $0,1$ desviaciones sería

$$z_N = 1,64 \tag{8.15}$$

por tanto,

$$\hat{d} < \mu_{\hat{d}} + 1,64\sigma_{\hat{d}} \tag{8.16}$$

Esto quiere decir que $1,64\sigma_{\hat{d}}$ de la media corresponden a un valor d con un nivel de confianza de $0,9$ (ie. al $90\,\%$) a ambos lados de la media, que equivalen a un nivel de 0,95 (ie. $95\,\%$) para un solo lado de la media, que es el valor que nos interesa. Así, en términos estadísticos podemos decir que aceptamos la hipótesis de que 8.12 en la página anteriorcon una confianza de $0,95$ o bien que rechazamos la hipótesis opuesta (hipótesis nula) con un nivel de significación de $0,05$ (ie. $1 - 0,95 = 0,05$).

8.1.5. Comparación de dos algoritmos de aprendizaje

Si deseamos comparar los resultados obtenidos por dos algoritmos de clasificación, C_1 y C_2, en vez de dos hipótesis, hemos de determinar si la diferencia observada entre ambos es significativa.

Tenemos que determinar cual de los dos algoritmos es mejor en promedio para una función objetivo dada, f. Este promedio tiene en cuenta la precisión relativa de ambos algoritmos sobre todos los conjuntos de tamaño n que se obtengan de una instancia de la distribución D. Este promedio es el valor esperado de la diferencia de errores:

$$E_{S \subset D}[error_D(C_1(S)) - error_D(C_2(S))] \tag{8.17}$$

Siendo $C(S)$ las hipótesis de salida (o resultados) del algoritmo de clasificación para un conjunto de datos S de entrenamiento, de forma que $S \subset D$.

Como usualmente las muestras han de tener un tamaño limitado, se dividen los datos en conjuntos disjuntos de entrenamiento (S_0) y prueba (T_0) y calculamos la media de los errores del conjunto de prueba para todos los experimentos. A la muestra completa se la denomina D_0;

$$\hat{\delta} \equiv \frac{1}{k} \sum_{i=1}^{k} \delta_i \tag{8.18}$$

Debemos tener al menos 20 o mejor, 30 ejemplos de entrenamiento para realizar esta técnica y que los resultados sean estadísticamente significativos.

$$\hat{\delta} \leftarrow error_{T_i}(h_1) - error_{T_i}(h_2) \tag{8.19}$$

De esta forma, $\hat{\delta}$ estima 8.17, siendo S una muestra aleatoria de D_0 y con tamaño $\frac{k-1}{k}|D_0|$. Con todo esto, podemos calcular el intervalo de confianza aproximado de $N\,\%$:

$$\hat{\delta} \pm t_{N,k-1} S_{\hat{\delta}} \tag{8.20}$$

$t_{N,k-1}$ es similar a z_N, de sus dos subíndices, el primero indica el nivel de confianza y el segundo los grados de libertad, denotado por v, que se refiere al

número de eventos aleatorios independientes que tienen que ver para producir el valor para la variable aleatoria, $\hat{\delta}$, en este caso $k-1$. Según $k \to \infty$, el valor de $t_{N,k-1}$ se acerca a la constante z_N.

Por su parte, $S_{\hat{\delta}}$ es un estimado de la desviación estándar de $\hat{\delta}$ tal que:

$$S_{\hat{\delta}} \equiv \sqrt{\frac{1}{k(k-1)} \sum_{i=1}^{k} (\delta - \hat{\delta})^2} \tag{8.21}$$

Si las muestras con que probamos a los algoritmos son idénticas, reciben el nombre de apareadas, dichas pruebas apareadas producen intervalos de confianza más ajustados porque las diferencias en errores se deben a los algoritmos y no a las diferencias de las muestras que se dan cuando no usamos muestras idénticas para los algoritmos.

En la prueba t-Student [9](o t-test en inglés) utilizada para comparar las diferencias de resultados para dos grupos, se verifica la diferencia entre las medias en relación con que cantidad varían los resultados individuales.

8.1.6. Matrices de confusión

Las matrices de confusión son una herramienta de visualización muy utilizada en aprendizaje supervisado. Cada columna de la matriz representa el número de predicciones de cada clase, mientras que cada fila representa a las instancias en la clase real, permitiendo ver si el algoritmo está confundiendo dos clases.

		Predecidos	
		Positivo	Negativo
Reales	Positivo	a	b
	Negativo	c	d

Cuadro 8.1: Matriz de confusión

A partir de dicha matriz podemos obtener la siguiente información:

- a: el número de veces que una instancia positiva ha sido clasificada como tal. No hay error, un verdadero positivo.
- b: el número de veces que una instancia negativa ha sido clasificada como positiva. Error estadístico de tipo II, falsos positivos.
- c: el número de veces que una instancia negativa ha sido clasificada como positiva. Error estadístico de tipo I, falsos negativos.
- d: el número de veces que una instancia negativa ha sido clasificada como negativa. No hay error, un verdadero negativo.

También existen una serie de términos habitualmente utilizados en estadística que se pueden obtener a partir de dichos datos:

- Exactitud (AC), proporción del total de predicciones correctas:

$$AC = \frac{a+d}{a+b+c+d} \tag{8.22}$$

- Tasa de verdaderos positivos (TP), proporción de positivos correctamente identificados:

$$TP = \frac{a}{a+b} \tag{8.23}$$

- Tasa de falsos positivos (FP), proporción de negativos incorrectamente clasificados como positivos:

$$FP = \frac{c}{c+d} \tag{8.24}$$

- Tasa de verdaderos negativos (TN), proporción de verdaderos negativos correctamente identificados:

$$TN = \frac{d}{c+d} \tag{8.25}$$

- Tasas de falsos negativos (FN), proporción de positivos incorrectamente clasificados como negativos:

$$FN = \frac{b}{a+b} \tag{8.26}$$

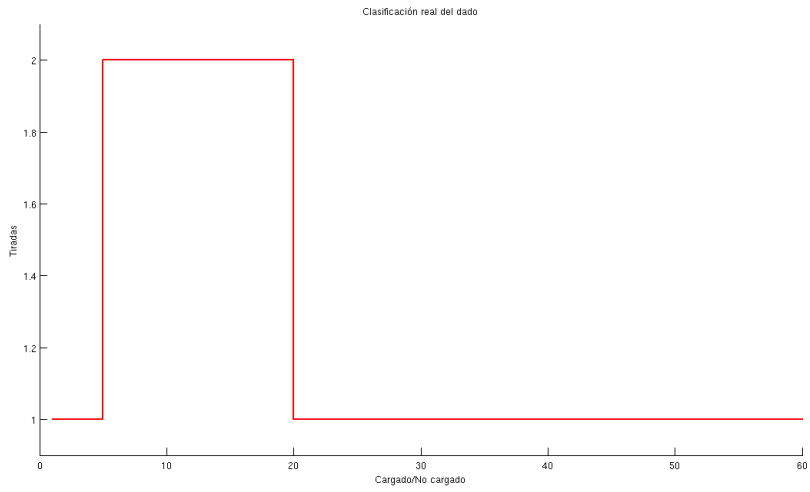
- La precisión (P), que es la proporción de positivos correctamente predichos:

$$P = \frac{a}{a+c} \tag{8.27}$$

Si bien habitualmente se utiliza un enfoque "clásico" que, vía matrices de confusión y teniendo en cuenta las diferencias de error entre múltiples hipótesis que se obtiene de la misma, permita calcular los resultados de dos clasificadores con determinado intervalo de confianza. Así y a modo de ejemplo, es posible calcular para la siguiente secuencia de tiradas de dados:

$$O_1 = [6, 2, 2, 3, 5, 6, 1, 6, 2, 2, 6, 6, 6, 6, 2, 6, 6, 6, 1, 1, 5, 3, 6, 3, 4, 1 \dots] \tag{8.28}$$

Que equivale a la siguiente clasificación de dados cargados (2: cargado, 1: no cargado):



y utilizando el modelo oculto de Markov que se muestra a continuación,

$$\lambda = \left(\begin{bmatrix} 0{,}95 \\ 0{,}05 \end{bmatrix}, \begin{bmatrix} 0{,}95 & 0{,}05 \\ 0{,}1 & 0{,}9 \end{bmatrix}, \begin{bmatrix} \frac{1}{6} & \frac{1}{6} & \frac{1}{6} & \frac{1}{6} & \frac{1}{6} & \frac{1}{6} \\ \frac{1}{10} & \frac{1}{10} & \frac{1}{10} & \frac{1}{10} & \frac{1}{10} & \frac{1}{2} \end{bmatrix} \right)$$

obtenemos como resultado que el HMM se adelanta y retrasa al clasificar los estados cargado / no cargado del dado:

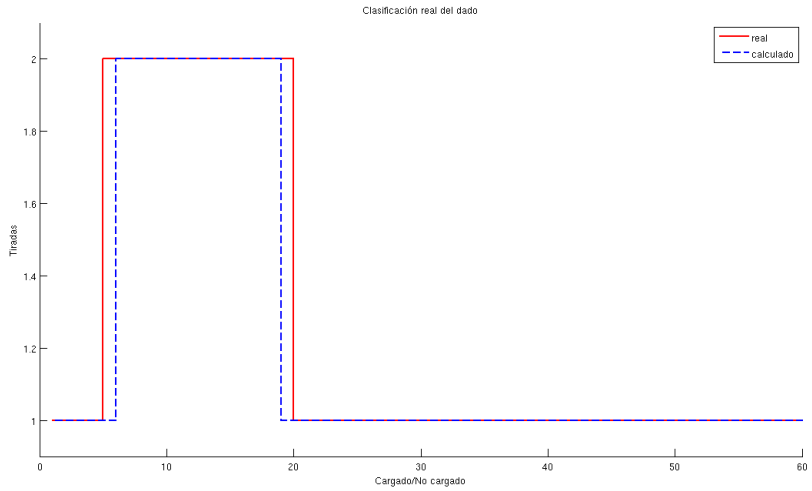


Figura 8.1: Clasificación del dado

Que nos da como matriz de confusión:

Matriz clásica

$$\begin{bmatrix} 47 & 4 \\ 0 & 9 \end{bmatrix}$$

Cuadro 8.2: Matriz de confusión

En este caso, vemos que la matriz de confusión está considerando los totales de fallos y aciertos, pero como veremos en 8.2 y la sección dedicada a resultados (ver V en la página 109), la matriz de confusión no tiene en cuenta ciertas peculiaridades del HMM, principalmente el hecho de que tiene memoria, con lo que se clasifican como errores ciertos estados que realmente se deben a retrasos o adelantos al reconocer secuencias.

8.2. Problemas en la aplicación de evaluadores clásicos a los HMM

Como se ha explicado en , debido al hecho de que los HMM tienen memoria, "tardan" uno o varios estados en detectar cambios en una secuencia:

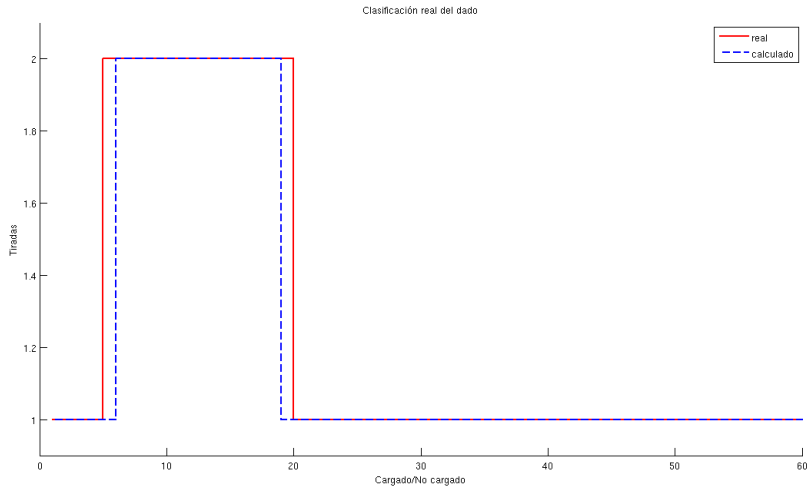


Figura 8.2: Adelantos y retrasos en detección

Debido a dicha inercia, el uso de métodos más clásicos para evaluar su funcionamiento puede darnos algunos resultados válidos de su comportamiento pero incompleta, dado que dichos métodos clásicos tienen en cuenta las tasas de totales, como verdaderos positivos, falsos positivos, verdaderos negativos y falsos negativos (8.1.6 en la página 81), pero no tienen en cuenta la posibilidad de que, cómo en la figura 8.2, el clasificador haya clasificado la secuencia de forma correcta pero con un retraso de uno o varios estados: en este caso, un método clásico está contando como dos errores lo que realmente es un pequeño retraso de un estado al detectar cambios. Si bien es cierto que se está dando un intervalo en el cual el clasificador "tarda" uno o varios estados en detectar el cambio, esto no nos supone un problema, puesto que lo que nos interesa es un clasificador que sea capaz de detectar que se ha producido cambio de estado a fin de segmentar la serie de forma correcta.

También puede darse el caso, en el que las probabilidades de transición estén poco balanceadas (el sistema es inestable y cambia continuamente de un estado a otro):

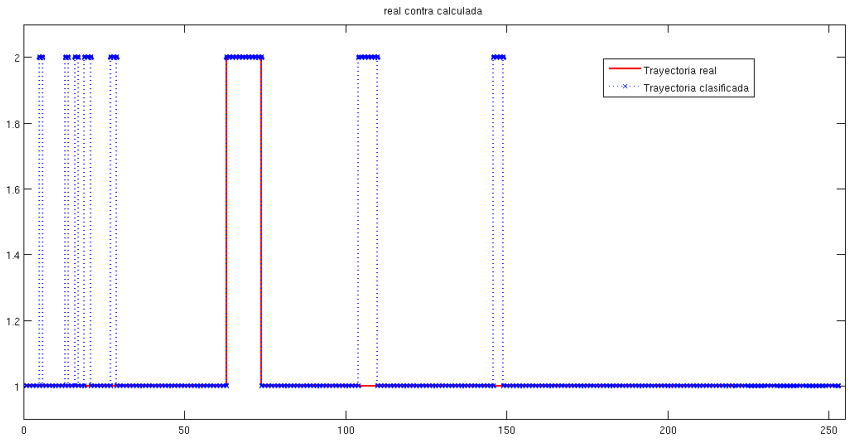


Figura 8.3: Sistema poco balanceado

En este caso, la tasa de errores del clasificador (exactitud y precisión, en 8.1.6 en la página 81) no es representativa del comportamiento del clasificador, por que debido a su inercia y por el propio diseño del clasificador, no detectará transiciones "cortas" en las que el sistema permanece durante muy poco tiempo, como las mostradas en la figura anterior (figura 8.3). Este caso, que puede parecer contradictorio con el explicado en el punto anterior, se da de forma habitual cuando queremos clasificar series temporales con ruido en las secuencias: nos interesa que el sistema trate como ruido y filtre los cambios muy cortos debido a que lo que nos interesa es que sea capaz de segmentar correctamente secuencias más largas. Esto es, deseamos que el sistema sea lo bastante sensible como para clasificar que en una partida como la mostrada en 8.2 en la página anterior, se ha utilizado un dado cargado, pero que no sea tan sensible como para clasificar como uso de dado cargado cada vez que aparece una secuencia con dos '6'.

Otro problema con el que nos hemos encontrado es el opuesto al que acabamos de explicar: que las probabilidades de transición estén poco balanceadas, los errores totales sean muy bajos a lo largo de la secuencia, y que nos interese que el clasificador los detecte, pero no lo haga. Obviamente, es necesario que la metodología tenga en cuenta este punto y el anterior, a fin de que seamos capaces de cambiar los parámetros del clasificador para balancearlo hacia un extremo u otro, en función de que nos interese más detectar pequeños cambios en secuencias con poco ruido, o ignorarlos en secuencias muy ruidosas.

Debido a estas razones, se necesita una metodología de evaluación más especifica que nos permita evaluar el comportamiento de HMM respecto a otros evaluadores a la hora de clasificar series temporales.

8.3. Análisis y evaluación de modelos

A fin de comparar de forma lo más correcta y completa posible el funcionamiento de los distintos evaluadores a utilizar para clasificar series temporales, tengan estos memoria o no (al tener en cuenta ambas posibilidades y creando una metodología homogénea, es posible comparar unos evaluadores con otros),

buscamos desarrollar un algoritmo que nos permita comparar dos clasificadores de características distintas entre sí.

Así, si deseamos saber cuál de los siguientes algoritmos, M_1 o M_2, ambos HMM es mejor a la hora de evaluar una secuencia de observaciones O_1, conociendo la secuencia real que la ha producido, S_1, a fin de, por ejemplo, entrenar un HMM (M_1 y M_2 serían el mismo HMM con diferentes grados de entrenamiento) o elegir un posible HMM para reconocer la secuencia (M_1 y M_2 serían dos HMM distintos) o incluso elegir distintos clasificadores.

Utilizando O_1 y el ya explicado algoritmo de Viterbi (ver 7.4 en la página 66), obtendríamos las secuencias calculadas:

$$Viterbi(M_1, O_1) \rightarrow S_1'$$

$$Viterbi(M_2, O_1) \rightarrow S_2'$$

Dados S_1'y S_2' y conociendo la secuencia real que los ha producido, S_1, obtendríamos las matrices de confusión de ambos modelos respecto a S_1para compararlos. En este caso, y como se ha explicado en el punto referente a las matrices de confusión (8.1.6 en la página 81) y trabajando con 'n' clases posibles, tendríamos n^2 elementos en la matriz.

8.3.1. Selección de Hipótesis

Se utiliza de forma general para cada posible modelo clasificador una de las siguientes hipótesis:

- H_0 : el modelo clasifica correctamente una observación O como perteneciente a la clase C.
- H_1 : el modelo no clasifica correctamente una observación O como perteneciente a la clase C.

La decisión, entre ambas posibles hipótesis se hace de acuerdo con el criterio de máxima verosimilitud (Maximum Likelihood, ML [10]), se obtiene mediante el cociente de verosimilitudes dado por:

$$\frac{\Pr(X|H_0)}{\Pr(X|H_1)} \begin{cases} \geq & \vartheta \, aceptar \, H_0 \\ < & \vartheta \, rechazar \, H_0 \end{cases} \tag{8.29}$$

donde $\Pr(X|H_i)$, $i = 0, 1$ es la probabilidad de la hipótesis H_i de acuerdo a un umbral de decisión determinado, representado por ϑ. Si dispusiéramos de información perfecta, ϑ debería ser 0, pero dado que no es posible dicho nivel de perfección, en la práctica se ajusta dicho umbral para controlar la relación entre las probabilidades de cometer errores en los dos sentidos posibles en la decisión.

Dado que nos interesa saber si un modelo está o no clasificando correctamente una serie temporal, utilizando las metodologías habituales, podemos tener como error

- Falso rechazo (FN), el modelo clasifica incorrectamente un positivo.
- Falso positivo (FP), el modelo clasifica incorrectamente un negativo.

Se suele elegir ϑ de forma que la tasa de falsos rechazos (tasa de FN, no detectados) y la de falsas detecciones (tasa de FP) se igualen en la medida de lo posible, en lo que estadísticamente viene a llamarse la EER (Equal Error Rate):

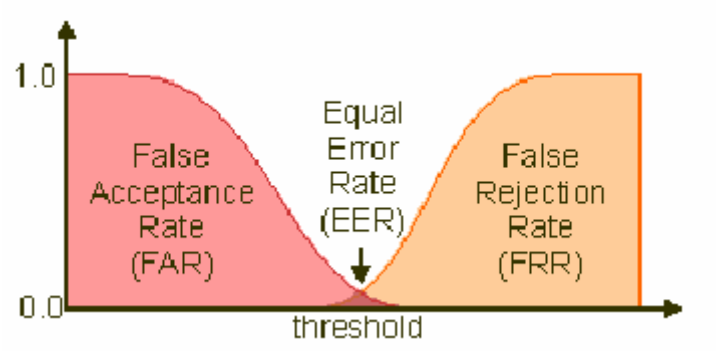


Figura 8.4: EER, Equal Error Rate. tasa de error equiprobable

De esta manera, es posible utilizar ϑ para afinar ambas tasas de manera que seamos capaces de trabajar en el punto en el cual se igualen las tasas de FN y FP. En dicho punto de intersección, es posible comparar distintos sistemas y es donde el error del sistema, dado como la suma de la FN y la FP, se suele minimizar.

El problema de este método de comparación es que para poder comparar dos sistemas según el EER, es necesario que éste sea calculado sobre los mismos datos de test utilizando el mismo protocolo experimental y además, requiere un gran número de pruebas. Otro problema es que el EER resulta de gran utilidad para afinar un modelo de manera que se igualen ambas tasas de error, pero no resulta útil para describir el rendimiento de un modelo en cuestión. En cambio, podría resultar de utilidad para futuros trabajos, especialmente para el entrenamiento de clasificadores.

8.3.2. Curvas ROC

Una curva ROC (acrónimo de Receiver Operating Characteristic, o Característica Operativa del Receptor [7] y [13]) es una representación gráfica de la sensibilidad frente al inverso de la especificidad para un sistema clasificador binario según varía el umbral de discriminación.

El análisis de la curva ROC, se utiliza para seleccionar modelos óptimos o probablemente óptimos y descartar modelos subóptimos independientemente de (y antes de especificar) el coste de la distribución de las dos clases sobre las que se decide.

Una de las ventajas de las curvas ROC es que son independientes de la distribución de las clases en la población y por tanto se pueden usar como test independientemente de la misma, no estando aplicadas a un problema o problemas en concreto.

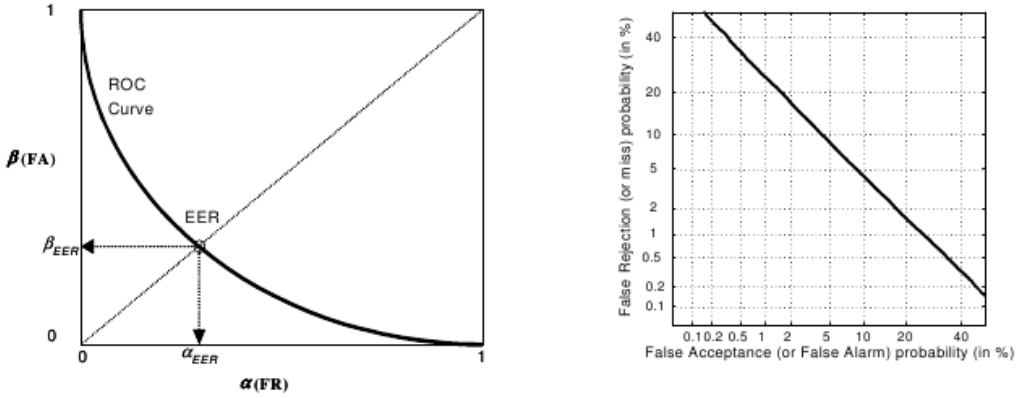


Figura 8.5: Curva ROC 1

La curva ROC se puede usar para generar estadísticos que resumen el rendimiento del clasificador, algunos de dichos estadísticos son:

1. El punto de intersección de la curva ROC con la línea perpendicular a la línea de no-discriminación.
2. El área entre la curva ROC y la línea de no-discriminación.
3. El área bajo la curva ROC, llamada comúnmente AUC (Area Under the Curve), que suele denominarse en la literatura como A'
4. Índice de sensibilidad o d' (d-prima, por cierto siempre minúscula). Es la distancia entre la media de la distribución de actividad en el sistema bajo condiciones de sólo ruido y su distribución bajo condiciones de sólo señal, dividido por su desviación típica, bajo el supuesto de que ambas distribuciones son normales con la misma desviación típica. Bajo estos supuestos, se puede probar que la forma de la curva ROC sólo depende de este parámetro d'.

De todos estos indicadores, el más utilizado es el área bajo la curva ROC o AUC. Este índice se puede interpretar como la probabilidad de que un clasificador ordenará o puntuará una instancia positiva elegida aleatoriamente más alta que una negativa. Además está demostrado que el área bajo la curva ROC es equivalente a otras pruebas estadísticas, como la Prueba de Mann-Whitney, la Prueba de los signos de Wilcoxon [11] o el Coeficiente de Gini [12], con la siguiente fórmula $G1 + 1 = 2 \cdot AUC$.

8.3.3. Curvas DET

Las curvas DET (Detection Error Tradeoff [7] y [13]) se obtienen a partir de las curvas ROC realizando una transformación no lineal en los ejes, transformando las curvas ROC en rectas o casi en rectas.

Esta transformación las hace más sencillas de analizar y comparar unas con otras, tanto de forma visual como automática.

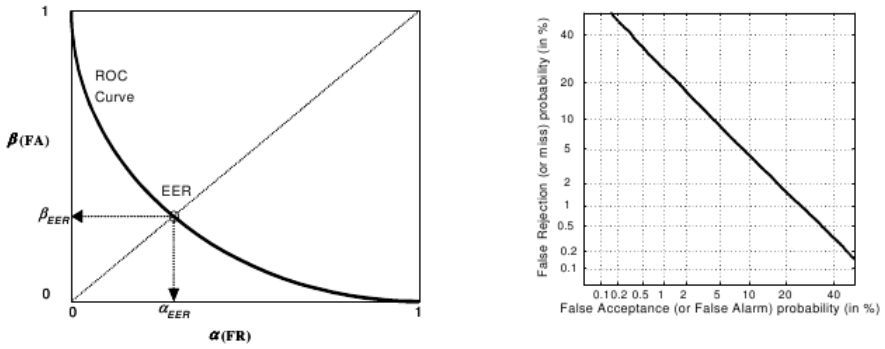


Figura 8.6: ROC vs. DET

Capítulo 9

Metodología de evaluación de modelos

Cómo se ha explicado a lo largo del 8 en la página 76, existen múltiples opciones a la hora evaluar un clasificador, sin embargo incluso las más completas, como las curvas ROC y DET, presentan problemas al utilizarse con clasificadores con memoria: dado el problema del retardo a la hora de reconocer las secuencias, explicado en 7.4 en la página 66, una técnica de evaluación que solo tenga en cuenta los ratios de error a la hora de evaluar un HMM será incompleta, ya que clasificará los retrasos como errores y no tendrá en cuenta el número de transiciones correctas que haya detectado el HMM.

Además, todas las técnicas estudiadas presentan problemas por que al solo tener en cuenta los ratios analizados, ignoran el número de cambios de estado que va teniendo el clasificador a lo largo del tiempo: como se discutirá en el 12.3 en la página 117, es posible que aún teniendo más aciertos, un clasificador sea menos fiable por ejecutar muchas transiciones falsas que otro más estable, como el HMM, que aunque tarde en reconocer un poco más un cambio de estado, no produce continuos cambios.

Como ejemplo, la siguiente secuencia devolvería dos errores, cuando el HMM ha detectado correctamente un cambio de estado, que era lo que nos interesaba detectar, pero con un pequeño retraso de dos estados para detectar el primer cambio, y otro adelanto de dos estados para volver a detectar el cambio:

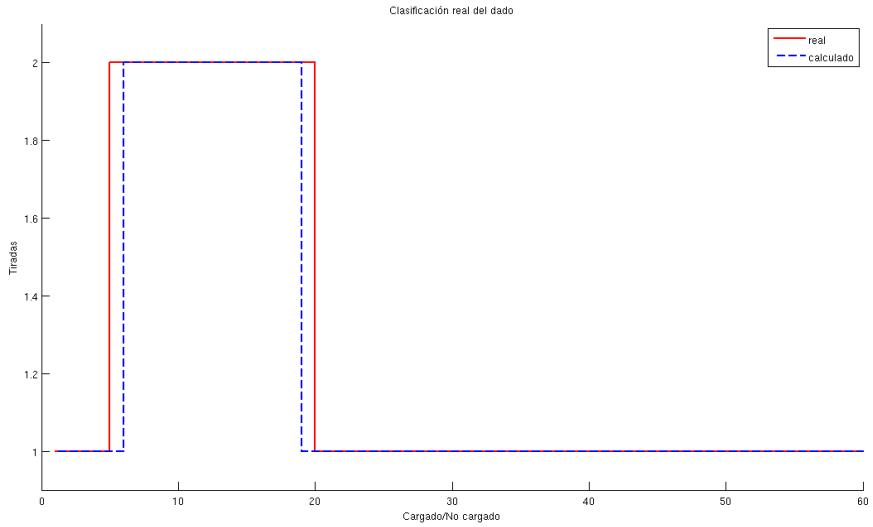


Figura 9.1: Retraso al reconocer la secuencia

A fin de evitar dichos problemas y para tener en cuenta todas las variables envueltas, usaremos una metodología que tenga en cuenta los siguientes puntos:

- Análisis de las transiciones entre estados.
- Comparar si los modelos son capaces de clasificar correctamente dichos cambios de estados.
- Obtención como resultado final de una matriz de confusión de orden superior.

A modo de ejemplo, utilizaremos en el problema de los dados cargados una secuencia calculada, S_1', obtenida por el clasificador a partir de una secuencia de observaciones O_1, pudiendo ser el algoritmo utilizado un HMM o un árbol C4.5 o cualquier otro clasificador; y la secuencia real que dio lugar a las observaciones, S.

9.1. Análisis de las transiciones entre estados.

En este primer paso, obtenemos el número total de cambios de estado en ambas secuencias, S_1' y S, obteniendo la derivada de sus envolventes.

Utilizaremos las derivadas de ambas porque nos interesan los cambios de estado y no los totales. A partir de dichas derivadas y por cada cambio de estado en la secuencia real, S, comprobaremos si se replica en S_1.

9.2. Comparar si los modelos son capaces de clasificar correctamente dichos cambios de estados.

En este punto es posible tener en cuenta la posibilidad de retrasos en la secuencia, de forma que nuestra herramienta prevé la posibilidad de realizar dicho análisis con un número máximo de retrasos o adelantos (explicado más detalladamente en el diseño de la herramienta, 11.3 en la página 107).

En el siguiente ejemplo, ya explicado en 8.2 en la página 83, una vez obtenidos los cambios totales de estado en ambas secuencias, podemos comparar si ambos modelos pueden clasificar dichos cambios de forma correcta, sin tener en cuenta los valores absolutos que nos inducían a los errores explicados en 8.2 en la página 83.

Como deseamos saber si el clasificador está reconociendo correctamente las transiciones y teniendo en cuenta el ya mencionado problema de los retrasos, compararemos ambas series, real y calculada; S y S'_1 de forma que se tenga en cuenta este hecho. Para facilitar el cálculo, utilizaremos una herramienta muy probada en el campo del procesamiento de la señal: la correlación cruzada, que mide las similaridades de dos señales en función de un determinado adelanto o retraso entre ellas.

El retraso puede ser tan largo como deseemos, pero a fin de facilitar las pruebas hemos realizado un análisis estadístico en los problemas a tratar para tener en cuenta de qué tamaño suelen ser los retrasos. En el caso del problema de los dados, los resultados del análisis son que en un 80 % de los casos, un HMM detecta el cambio en un máximo de cinco estados, así que en este caso, la herramienta clasificará como retrasos los cambios que se produzcan en un intervalo máximo de 5 estados y como error los que sean mayores.

Como dato importante y aunque siempre se mencionen como retrasos, es posible que el HMM reconozca un cambio antes de que se produzca: esto se debe a la naturaleza probabilística del algoritmo de Viterbi en algunos casos. Por esa razón, se tienen en cuenta adelantos en dicho intervalo.

En el caso de las trayectorias, los resultados de dicho análisis se mostrarán en la parte dedicada a los resultados experimentales, en V en la página 109.

Por supuesto, el análisis de los retrasos se puede obviar en el caso de clasificadores sin memoria (se hacen los análisis con un retraso/adelanto posible de tamaño cero).

9.3. Obtención como resultado final de una matriz de confusión de orden superior.

Con el fin de desarrollar una metodología completa de evaluación de modelos capaz de ofrecer resultados para modelos sin y con memoria, hemos utilizado una matriz de confusión de orden superior, que en lugar de centrarse en las diferencias entre las clases real y predichas, analiza las transiciones entre estados en ambas secuencias a fin de comprobar si dichas transiciones son o no detectadas por el algoritmo de clasificación.

De esta forma, dicha matriz de orden superior nos permite analizar el total de falsas transiciones, debidas a ruido en las transiciones, verdaderos positivos

y retrasos en la detección de transiciones debidos a la naturaleza del HMM.

Dado que es equivalente a la matriz clásica de confusión, se puede utilizar también con modelos sin memoria, por lo que será utilizada para hacer las comparaciones de rendimiento frente a otros clasificadores.

Esta matriz estaría compuesta de:

- Estados correspondientes a las transiciones, T
- Falsas transiciones, esto es, los estados en los cuales no hubo un cambio real de estado pero el modelo utilizado detectó un cambio, FT.
- Transiciones no detectadas, en los cuales sí hubo un cambio real de estado que no fue detectado por el modelo. ND.
- Sin transición, ni hubo cambio ni fue detectado. NT

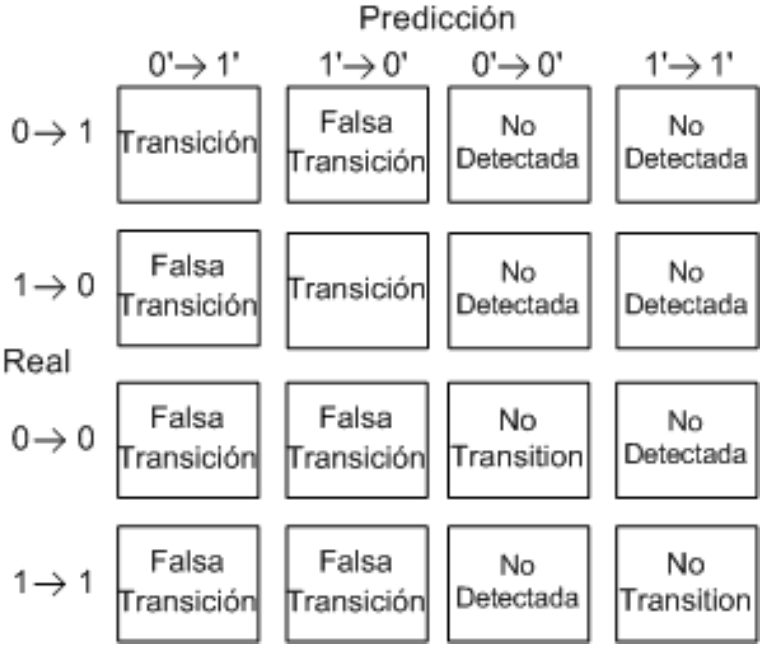


Figura 9.2: Matriz de confusión de orden superior

Sin embargo, dado que esta matriz requiere tener en cuenta todas las posibles transiciones de estados, incluyendo el caso de no haya un cambio de estado por cada posible pareja de estados, lo cual hace que sea necesaria una matriz de tamaño n^4, siendo n el número de posibles hipótesis, que en nuestro caso son dos: pertenece o no a una clase dada , dando como resultado una matriz de 2^3.

Para facilitar los cálculos y reducir la complejidad, se ha simplificado añadiendo un estado de no transición (llamado NT), en el cual se tienen en cuenta todos los posibles casos en los cuales no ha habido un cambio de estado. La fila correspondiente a este estado contendrá información relativa a estados sin transición, mientras que la columna correspondiente, contendrá información relativa a transiciones no detectadas. De esta forma, la matriz simplificada se reduce a:

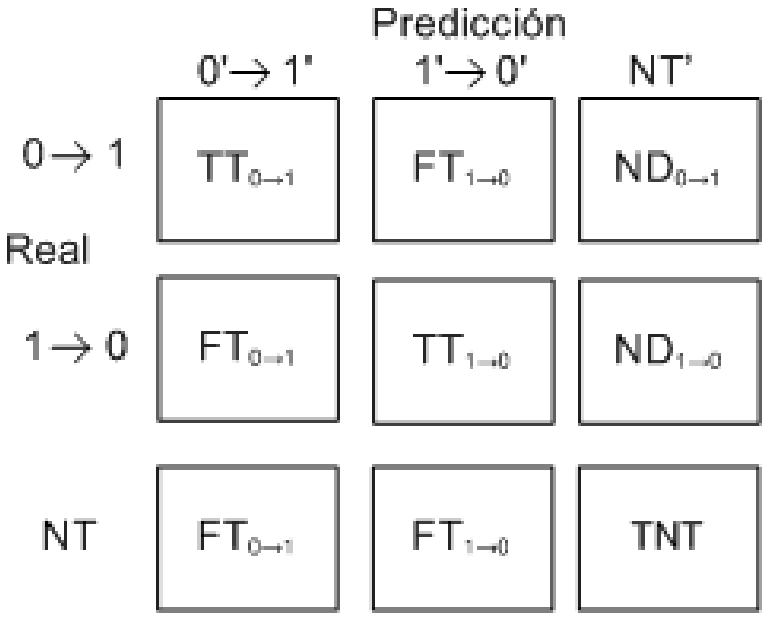


Figura 9.3: Matriz de confusión de orden superior (simplificada)

Dónde:

- TT: transición existente y detectada.
- ND: transición existente y no detectada.
- FT: falsa transición.
- TN: transición no existente.
- TNT, verdadera no transición

Parte IV

Herramienta para la Clasificación de Series Temporales y de Evaluación de Clasificadores

Como se ha indicado en el epígrafe dedicado a los objetivos en 2 en la página 11, deseamos una herramienta capaz de:

- Clasificar series temporales en base a Modelos Ocultos de Markov.
- Evaluar clasificadores.

Al poder dividirse realmente en dos herramientas con objetivos diferenciados, se explicará su diseño e implementación en dos epígrafes distintos para que el lector o usuario pueda leer la parte correspondiente a la herramienta que sea de su interés.

También se incluirá tanto un manual de usuario como apéndice y una API (*Abstract Programming Interface*) para su uso por parte de desarrolladores en el apéndice A en la página 130.

Capítulo 10

Herramienta de Clasificación de Series Temporales con Modelos ocultos de Markov

Esta herramienta se ha desarrollado con el software de desarrollo matemático *Matlab* [14] y su lenguaje de programación, *m*, por ser un entorno de desarrollo muy utilizado en los ámbitos académicos y de ingeniería y sobre todo por que al estar enfocado al uso de rutinas matemáticas y trabajo con vectores y matrices facilita un entorno para el desarrollo y pruebas de programas que ejecuten los algoritmos requeridos para esta herramienta de clasificación.

Al ser un lenguaje muy utilizado en el ámbito académico, existen paquetes de herramientas (*toolboxes*) que implementan diversas funcionalidades y algoritmos matemáticos. Uno de dichos paquetes de herramientas, el *Murphy HMM toolbox [15]* es el que se ha utilizado como implementación de la funcionalidad de los Modelos Ocultos de Markov.

Aunque también existe una implementación de los HMM en el paquete de herramientas [14], se ha elegido el diseñado por *Murphy* [15] por varias razones (algunas de las cuales se explicarán con más detalle en 10.1 en la página siguiente):

- El paquete *Murphy HMM toolbox* ofrece la funcionalidad de trabajar con HMM en tiempo continuo (7.8 en la página 73) o discreto, mientras que su homónimo de *Matlab* sólo permite el uso de HMM en tiempo discreto. Dado que nuestro clasificador de trayectorias tiene como entrada una continua, esta ha sido una de las razones de más peso para elegirlo.
- El paquete *Murphy HMM toolbox* está optimizado para reducir uso de memoria y de procesador, al utilizar los algoritmos descritos en 7.7 en la página 71
- Ofrece en un único paquete múltiples herramientas de utilidad para el uso de HMM, como pueden ser las descritas en 7.5.3 en la página 70, 7.6.1 en la página 70 y otras que no se encuentran en el paquete *statistical toolbox* de *Matlab*.

- Es un paquete probado y que lleva siendo utilizado más tiempo que otros más recientes.

10.1. Paquete HMM Murphy toolbox

Este paquete se utiliza para proporcionar la funcionalidad básica para utilizar HMM tanto en tiempos discretos como continuo, así como otras funcionalidades estadísticas necesarias al utilizar HMM. Presentaremos una breve descripción de las características de la herramienta y la funcionalidad ofrecida, así como los problemas encontrados al con cada una de las funciones que se han utilizado.

Siguiendo el orden de explicaciones dadas en la memoria, se presentarán primero las funciones discretas y después las continuas.

10.1.1. Entrenamiento discreto

La función *dhmm_em* se utiliza para entrenar una modelo discreto con el algoritmo de EM (realmente utiliza el algoritmo Baum-Welch, que es una simplificación del EM [4]). Esta función recibe como parámetros un modelo oculto de Markov discreto, formado por un vector con las probabilidades iniciales del modelo, una matriz de probabilidades de transición, una matriz de probabilidades de observación y un conjunto de datos de entrenamiento e intenta entrenar el modelo que recibe como parámetro, para mejorar la capacidad del mismo para reconocer dichas secuencias.

Como resultado, produce un modelo mejorado y una matriz logarítmica de probabilidades, que se puede utilizar para comprobar si entre sucesivas iteraciones del algoritmo se mejora la probabilidad de observar una determinada cadena de entrenamiento.

El principal problema encontrado con este algoritmo es que por su propia definición [4] se trata de un algoritmo de búsqueda local, con cierta tendencia a devolver máximos locales como resultado, con lo cual debe tenerse en cuenta que los puntos desde los que se parte el entrenamiento son de gran importancia, así en el caso siguiente,

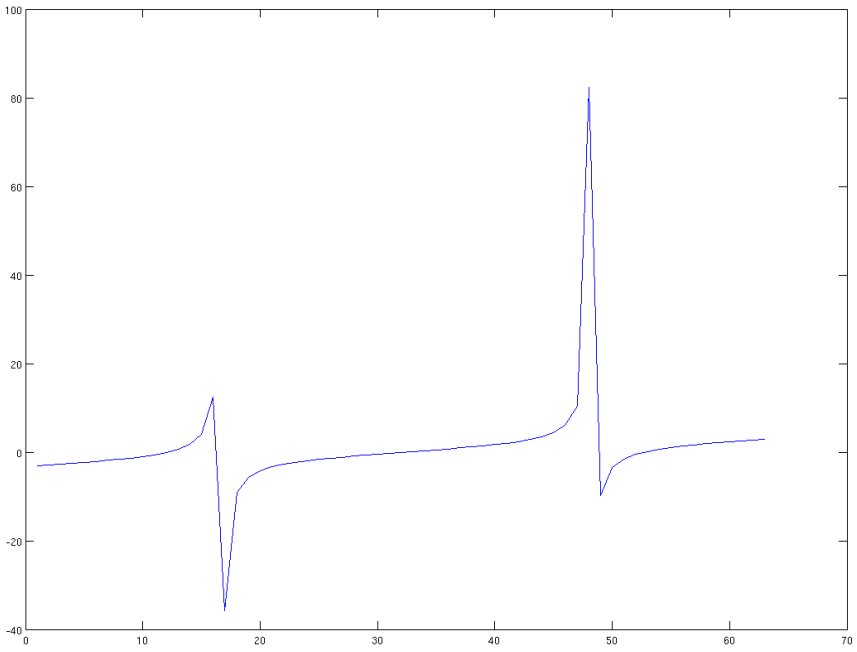


Figura 10.1: Máximo local y global

los resultados del algoritmo serán distintos si los valores iniciales del HMM hacen que este caiga cerca del valor '10', donde hay un máximo local, que aunque pueda ser bueno y clasifique de forma correcta, no será tan eficaz como si los valores iniciales están cerca del máximo global, en '40'.

Quizá sería recomendable en futuras investigaciones utilizar algún algoritmo de búsqueda global, como podría ser el *simmulated annealing* [16] u otro similar, que permita primero hacer una búsqueda global para encontrar buenos puntos iniciales, para después finalizar con una búsqueda local con Baum-Welch.

10.1.2. Entrenamiento continuo

El entrenamiento continuo es bastante similar al discreto, salvo que al tratarse de HMM continuos, en lugar de utilizarse una matriz de probabilidades de observación, se utilizan funciones gaussianas de densidad de probabilidad para representar las probabilidades de las variables ocultas.

Estas gaussianas vienen dadas por dos parámetros, la media (μ) y la varianza (σ), de forma que la llamada a la función de entrenamiento continuo, *gausshmm_train_observed*, se hace tomando como parámetros los datos de entrenamiento, las matrices de probabilidad inicial y de transición, y los valores con la media y varianza de la gaussiana.

Como resultado, devuelve un modelo mejorado y una matriz logarítmica de probabilidades, que se puede utilizar para comprobar si entre sucesivas iteraciones del algoritmo, mejora la probabilidad de observar una determinada cadena

de entrenamiento.

Al igual que con su variante discreta, esta función está basada en el algoritmo EM simplificado (Baum-Welch), de forma que se trata de una función de búsqueda local, por lo que en caso de encontrar máximos locales al entrenar un HMM los devolverá como resultado válido. Es por tanto muy dependiente de los valores iniciales que reciba como parámetro. Además existe una patología agravada: es posible que en determinados puntos la varianza de unos datos al aplicar el HMM tienda a 0, dando como resultado una probabilidad infinita, hecho que de ocurrir, suele darse cerca del máximo global (esto es, el modelo más perfecto para esa secuencia dada).

En muchos casos se parte del hecho de que de por sí solo, el algoritmo EM no es capaz de encontrar dicho punto para evitar dicha patología. De hecho esta función no la tiene en cuenta y en las pruebas realizados no se ha dado el caso, aunque no puede descartarse que en futuras pruebas se encontrara.

Sin embargo, es necesario tener en cuenta que podría darse dicho caso y que se daría justamente en la mejor solución posible.

Al entrenar un HMM con este algoritmo, existen otros problemas que debemos tener en cuenta:

- Disminución de la probabilidad logarítmica durante el entrenamiento: este caso sugiere que uno de los componentes de la mezcla no ofrece suficientes datos, lo que indica que sería mejor probar con menos clases de datos o bien que el HMM recibido es poco representativo (habría que hacer una mejor inicialización).

- Matriz de varianza singular (determinante de σ sea 0 con distribuciones multivariadas): puede darse en mezclas multivariadas, se puede dar con una muestra de datos pequeña o con una inicialización incorrecta del modelo (concretamente, de las medias μ). Puede arreglarse si se usa una mejor inicialización o se especifica como parámetro que la covarianza sea diagonal o esférica. Dado que los modelos utilizados en este proyecto no eran multivariados, no se ha observado este problema.

El principal problema encontrado con este algoritmo es que por su propia definición [4] se trata de un algoritmo de búsqueda local, con cierta tendencia a devolver máximos locales como resultado, con lo cual debe tenerse en cuenta que los puntos desde los que se parte el entrenamiento son de gran importancia, así en el caso siguiente,

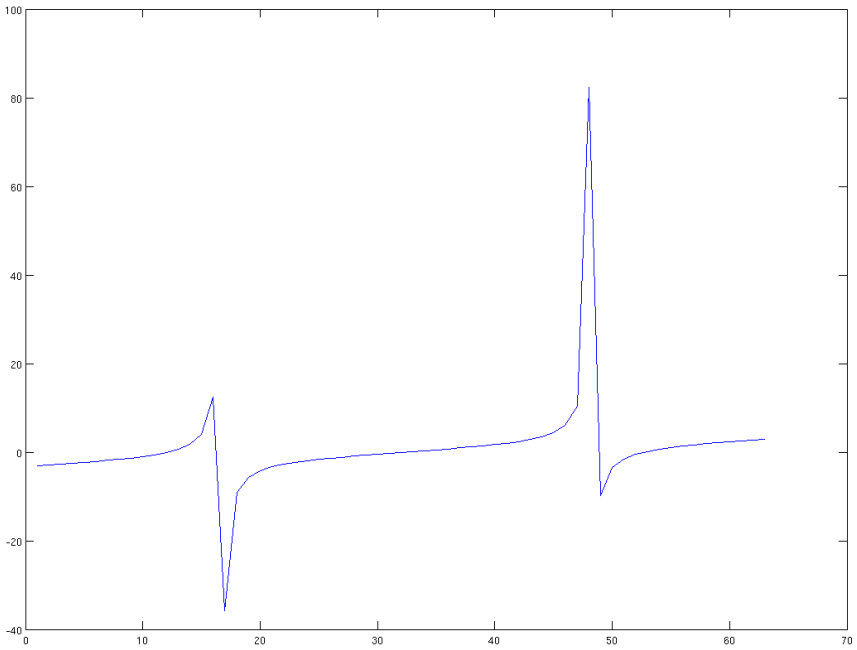


Figura 10.2: Máximo local y global

los resultados del algoritmo serán distintos si los valores iniciales del HMM hacen que este caiga cerca del valor '10', donde hay un máximo local, que aunque pueda ser bueno y clasifique de forma correcta, no será tan eficaz como si los valores iniciales están cerca del máximo global, en '40'.

Quizá sería recomendable en futuras investigaciones utilizar algún algoritmo de búsqueda global, como podría ser el simmulated annealing [16] u otro similar, que permita primero hacer una búsqueda global para encontrar buenos puntos iniciales, para después finalizar con una búsqueda local con Baum-Welch.

10.1.3. Clasificación de secuencias dado un HMM discreto

La función dhmm_logprob se puede utilizar para clasificar una secuencia dado un modelo, devolviendo como resultado la probabilidad logarítmica de que en un HMM se de un determinado conjunto de observaciones.

Se utiliza cuando se tienen varios HMM candidatos a generar una determinada secuencia y se quiere saber cual de todos ellos es el que tiene más probabilidades de reconocerla.

También resulta de utilidad para entrenamientos y evaluación de modelos ocultos de Markov por que genera un coeficiente, dados un modelo y un conjunto de observaciones, de forma que permite dibujar una curva de aprendizaje para una secuencia dada y sucesivos entrenamientos de un modelo.

Si bien es muy útil para evaluar HMM de esta manera, al no tener en cuenta toda la funcionalidad que era necesaria para evaluación de clasificadores descrita

en 8 en la página 76, se optó por desarrollar una metodología de evaluación, 9 en la página 90. En cualquier caso, esta función se puede utilizar para el entrenamiento de HMM y de hecho es utilizada por las funciones de entrenamiento discreto.

Recibe como parámetros una secuencia de observaciones, y las matrices de probabilidades inicial, de transición y de observación, devolviendo la probabilidad de que el modelo dado haya generado dicha secuencia de observaciones.

Los datos se asume que pertenecen a un alfabeto $\{1, 2 \dots S\}$, siendo S el tamaño de la matriz de transición recibido. No pueden contener '0'.

10.1.4. Clasificación de secuencias dado un HMM continuo

La función *mhmm_ logprob* es muy similar a *dhmm_ logprob*: se puede utilizar para clasificar una secuencia dado un modelo, devolviendo como resultado la probabilidad logarítmica de que en un HMM se dé un determinado conjunto de observaciones.

Al igual que su variante discreta, se puede utilizar cuando se tienen varios HMM candidatos a generar una determinada secuencia y se quiere saber cual de todos ellos es el que tiene más probabilidades de reconocerla, o para entrenamientos y evaluación de modelos ocultos de Markov, al generar un coeficiente dados un modelo y un conjunto de observaciones, permitiendo dibujar una curva de aprendizaje para una secuencia dada y sucesivos entrenamientos de un modelo.

Si bien es muy útil para evaluar HMM de esta manera, al no tener en cuenta toda la funcionalidad que era necesaria para evaluación de clasificadores descrita en 8 en la página 76, se optó por desarrollar una metodología de evaluación, 9 en la página 90. En cualquier caso, esta función se puede utilizar para el entrenamiento de HMM y de hecho es utilizada por las funciones de entrenamiento discreto.

Recibe como parámetros una secuencia de observaciones, y las matrices de probabilidades inicial y de transición, así como los valores que definen las gaussianas de emisión de probabilidades ocultas, μ y σ, las medias y varianzas. Devuelve como resultado la probabilidad de que el modelo dado haya generado dicha secuencia de observaciones.

Los datos se asume que pertenecen a un alfabeto $\{1, 2 \dots S\}$, siendo S el tamaño de la matriz de transición recibido. No pueden contener '0'.

10.1.5. Generar una secuencia de observaciones de prueba dado un HMM discreto

Se puede obtener una secuencia de observaciones que incluya datos observados y las transiciones ocultas mediante la función *dhmm_ sample*, que recibe como parámetros las matrices que definen un HMM discreto así como las longitudes de la secuencia y el número de secuencias a generar basados en estos parámetros. Este último puede ser necesario porque en determinados casos son necesarias múltiples secuencias de observaciones, 7.5.3 en la página 70.

Se puede utilizar para generar secuencias artificiales de pruebas que incluyen además de la observación las transiciones ocultas, de forma que facilite los entrenamientos o las evaluaciones de los HMM.

Recibe como parámetros las longitudes de secuencia y número de secuencias a generar, cuando haga falta generar múltiples secuencias de observación, así como las matrices que definen el modelo de Markov: probabilidades inicial, de transición y de observación.

Como resultado, genera una secuencia de valores observados y una secuencia de los valores ocultos que han dado lugar a dichas observaciones.

10.1.6. Generar una secuencia de observaciones de prueba dado un HMM continuo

Al igual que con el modelo discreto, se puede obtener una secuencia de observaciones que incluya datos observados y las transiciones ocultas mediante la función mhmm_sample.

Se puede utilizar para generar secuencias artificiales de pruebas que incluyen además de la observación las transiciones ocultas, de forma que facilite los entrenamientos o las evaluaciones de los HMM.

Recibe como parámetros las longitudes de secuencia y número de secuencias a generar (cuando haga falta generar múltiples secuencias de observación), así como las matrices probabilidades inicial y de transición y los valores que definen las gaussianas de emisión de probabilidades ocultas, μ y σ, las medias y varianzas.

Como resultado, genera una secuencia de valores observados y una secuencia de los valores ocultos que han dado lugar a dichas observaciones.

10.1.7. Probabilidad de que una secuencia haya sido generada por un HMM discreto

Aunque no se suele utilizar directamente, esta función es de gran importancia en el algoritmo de Viterbi que se verá en 10.1.8: la función *fwdback* calcula la probabilidad de que una secuencia de observaciones sea generada por un modelo dado.

Hay que resaltar que para reducir el uso de memoria y de tiempo de proceso se ha hecho uso de la propiedad descrita en 7.7 en la página 71.

Recibe como entrada una matriz de observaciones condicionadas, calculada de forma similar para las variantes discreta y continua, pero que devuelve como resultado una matriz similar para ambos casos, de forma que las llamadas a esta función no tienen diferencia cuando se usan con HMM discretos o continuos.

10.1.8. Cálculo de la secuencia oculta más probable

Se utiliza la función *viterbi_path*, que dadas las matrices de probabilidad inicial y de transición así como una matriz de observaciones condicionadas dada una entrada calculada como:

$$obslik(i,t) = \Pr(X_t|S_t = i)\nabla t, i \tag{10.1}$$

Generada por las funciones ofrecidas por el *HMM Murphy Toolbox* para el efecto (*multinomial_prob* en la variante discreta y *mixgauss_prob* en la continua) para sus variantes discreta y continua. al utilizarse dicha función, se

obtienen las probabilidades condicionadas, pudiendo usarse *viterbi_path* para obtener el camino más probable tanto con HMM discretos como continuos.

La función v*iterbi_path* devuelve el camino más probable generado, utilizando la función *fwdbac* para calcular las distintas probabilidades, estando optimizada de forma que reduce el uso de memoria y aumenta su velocidad respecto a las librerías ofrecidas por *Matlab*.

La principal diferencia entre las variantes discreta y continua es que la primera, calcula las matrices de observación con una función de cálculo multinomial discreto (*multinomial_prob*) y la segunda, las calcula con una mezcla de gaussianas continua (*mixgauss_prob*).

La función también presenta diferencias cuando se utiliza con una o varias secuencias a clasificar, en concreto cuando se utiliza con una única secuencia, debe llamarse con un vector de datos en forma de columna, mientras que si hay varias, se llama con una matriz con tantas filas como datos.

Para simplificar las llamadas se han desarrollado funciones que las realizan de forma automática en base a si son o no continuas.

10.2. Simplificaciones de las llamadas al paquete HMM Murphy toolbox

Dado que las llamadas al paquete HMM Murphy toolbox son, en muchos casos, poco intuitivas y no se dispone de documentación que las explique, se han creado funciones que sirven como envoltorio, y facilitan las llamadas a las funcionalidades más utilizadas. Concretamente a las clasificaciones de secuencia dado un HMM y al cálculo de la secuencia oculta más probable.

10.2.1. Cálculo de la secuencia oculta más probable

La función *viterbi* se utiliza para llamar a la función *viterbi_path*, teniendo dos posibles llamadas:

- Discreta: recibe como parámetros las matrices de probabilidad inicial (π, 5 en la página 57), de transición (A, 7.28 en la página 57), de observación (B, 7.30 en la página 57) y la secuencia de datos de observación (O, 7.33 en la página 57), en formato vector o matriz.

- Continua, similar a la anterior, pero como entrada: probabilidades iniciales (π, 5 en la página 57), probabilidades de transición (A, 7.28 en la página 57), media de las gaussianas (μ, 7.8 en la página 74), varianza de las gaussianas (σ, 7.8 en la página 74) y datos observados (O, 7.33 en la página 57), en formato vector o matriz. También se puede especificar un vector de ponderación (c, 7.8 en la página 74), de no especificarse, se da el mismo valor de ponderación a todas las gaussianas a mezclar.

Dependiendo de estos parámetros, se encarga de calcular la matriz de observaciones condicionadas con llamadas a *multinominal_prob* o *mixgauss_prob*, y de llamar a la función que implementa el cálculo de la secuencia con el algoritmo de viterbi, devolviendo dicha secuencia como resultado.

Capítulo 11

Herramienta de Evaluación de Clasificadores

Esta herramienta implementa la metodología de evaluación indicada en 9 en la página 90, para esto se dispone de varias funciones, la principal es la encargada de generar una matriz de orden 2, como la indicada en 11.3 en la página 107 que calcula la matriz de orden superior:

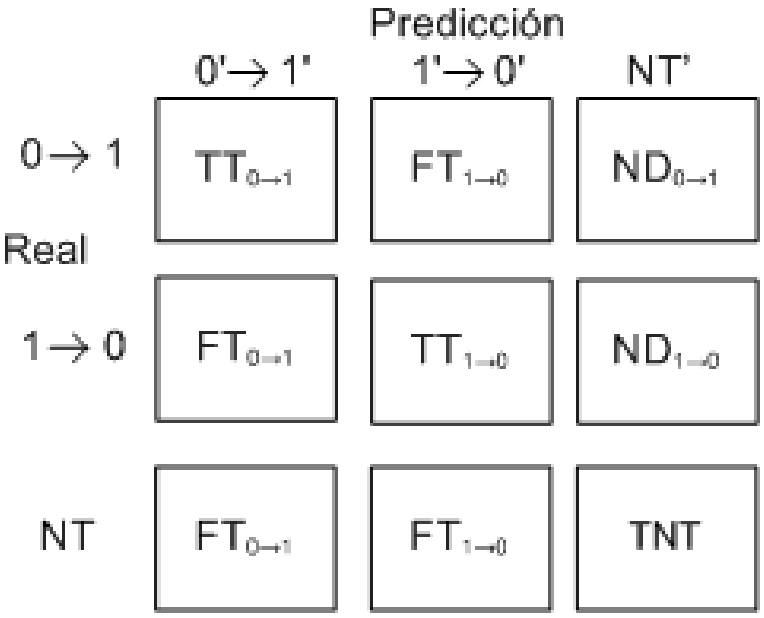


Figura 11.1: Matriz de confusión de orden superior

que tenga en cuenta los posibles retrasos o adelantos que pueda tener un HMM al reconocer una determinada secuencia.

También se ha implementado una función que genera una matriz de confusión clásica a fin de comparar los resultados de ambas metodologías.

11.1. Matriz de Confusión

La función *matriz_ confusión* genera una matriz de confusión, 8.1.6 en la página 81, a partir de una secuencia calculada por un clasificador y de la secuencia real que debería haberse calculado. A partir de dicha matriz de tipo:

		Predecidos	
		Positivo	Negativo
Reales	Positivo	a	b
	Negativo	c	d

Cuadro 11.1: Matriz de confusión

Es posible calcular los siguientes términos:

- Exactitud (AC), proporción del total de predicciones correctas:

$$AC = \frac{a+d}{a+b+c+d} \tag{11.1}$$

- Tasa de verdaderos positivos (TP), proporción de positivos correctamente identificados:

$$TP = \frac{a}{a+b} \tag{11.2}$$

- Tasa de falsos positivos (FP), proporción de negativos incorrectamente clasificados como positivos:

$$FP = \frac{c}{c+d} \tag{11.3}$$

- Tasa de verdaderos negativos (TN), proporción de verdaderos negativos correctamente identificados:

$$TN = \frac{d}{c+d} \tag{11.4}$$

- Tasas de falsos negativos (FN), proporción de positivos incorrectamente clasificados como negativos:

$$FN = \frac{b}{a+b} \tag{11.5}$$

- La precisión (P), que es la proporción de positivos correctamente predichos:

$$P = \frac{a}{a+c} \tag{11.6}$$

La función matriz_confusión genera a partir de las secuencias de datos real y calculada, y de los estados que existen (en los casos probados en este proyecto, se utiliza [1 2] para los estados 'estable' y 'maniobra', respectivamente) genera la matriz de confusión. Es posible calcular los términos indicados previamente con la función términos_matriz_confusion

11.2. Términos Calculados a partir de la Matriz de Confusión

La función terminos_matriz_confusion se utiliza para calcular la exactitud (AC), tasa de verdaderos positivos (TP), tasa de falsos positivos (FP), tasa de verdaderos negativos (TN), tasa de falsos negativos (FN) y la precisión (P) del algoritmo utilizado a partir de la matriz de confusión cuyos términos se desea calcular.

11.3. Matriz de Confusión de Orden Superior

La función *matriz_confusion_orden2* calcula la matriz de confusión de orden superior indicada en 11.3. Esta matriz esta compuesta de:

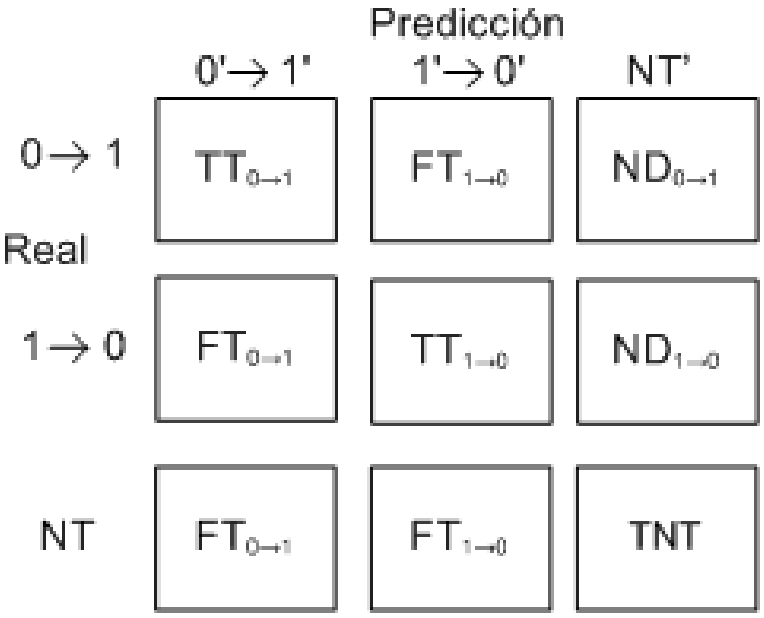


Figura 11.2: Matriz de confusión de orden superior

Dónde:

- TT: transición existente y detectada.
- ND: transición existente y no detectada.
- FT: falsa transición.
- NT: transición no existente.
- TNT, verdadera no transición

Para calcular dicha matriz se parte de dos secuencias de datos: la que contiene la clasificación real, la calculada, los estados; y el retraso máximo con el que se

quiere trabajar. El valor del retraso máximo no resulta de utilidad en algoritmos que no tienen tendencia a retrasarse o adelantarse a un cambio de estado, de hecho su uso en estos casos sería contraproducente por que enmascararía algunos errores.

Por ejemplo y como se ha discutido en 8.2 en la página 83, al tener memoria el HMM suele tener un pequeño retraso al detectar cambios de estado. Además en algunas ocasiones, la naturaleza no determinista del algoritmo de Viterbi (7.1.1 en la página 49) hace que se adelante.

Con el valor del retraso máximo, se le permite al algoritmo buscar los retrasos o adelantos en una ventana de tamaño $\pm retraso\,maximo$. La búsqueda comienza en el inicio de la transición real no detectada y termina al final de la ventana, no permitiendo que transiciones ya clasificadas, se puedan tomar como transiciones retrasadas o adelantadas.

Los valores del retraso máximo dependerán de las secuencias a considerar. Cuanto mayores sean dichos valores más posibilidades hay de que fallos al detectar una transición, se consideren un retraso o para valores pequeños, clasifiquen como error un retraso grande.

Un ejemplo lo podemos ver en esta transición, explicada también en el resultado experimental 12.3:

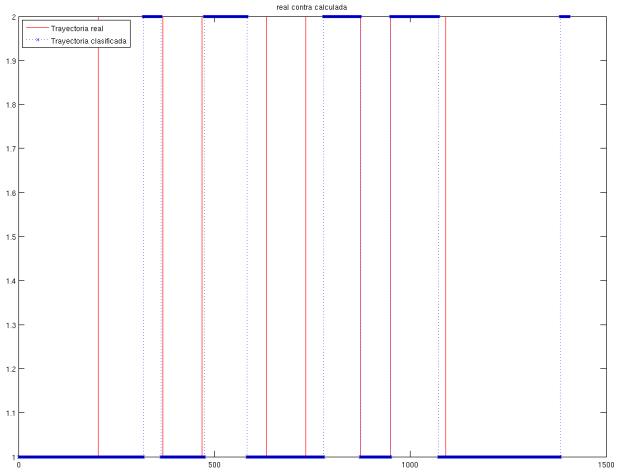


Figura 11.3: Retrasos en la detección

Con un valor grande de retraso, se clasifica como transición correcta la primera transición, sin embargo, si se reduce el valor del retraso, la primera transición se considera errónea.

Para evitar problemas relacionados con los retrasos o adelantos, la herramienta solo tiene en cuenta cada transición una vez por retraso: una vez se ha contado éste, no se vuelve a tener en cuenta para evitar posibles errores, y si de dos transiciones, la primera se considera errónea y la segunda válida, aunque la primera entre dentro del rango de una tercera por adelanto o retraso, no podrá tenerse en cuenta.

Parte V

Resultados Experimentales

A continuación se mostrarán algunos resultados del clasificador, para mostrar las ventajas de los sistemas basados en HMM frente a otros algoritmos como los árboles binarios, tanto desde el punto de vista del rendimiento de ambos modelos de clasificación, como para presentar el funcionamiento de la herramienta desarrollada y de la metodología de evaluación.

Para facilitar la revisión de las pruebas se indicarán las trayectorias utilizadas y los HMM y árboles generados para la clasificación de las mismas, de tal modo que se puedan utilizar como entrada para la herramienta desarrollada.

Se mostrarán para todas las pruebas diagramas conteniendo:

- Una representación de la trayectoria que permita al lector hacerse una idea de los problemas que puedan tener los algoritmos para reconocerlos.
- Una representación de los residuos de la trayectoria, que permita conocer el nivel de ruido de la misma.
- La clasificación obtenida por ambos clasificadores frente a la real.
- Los resultados de las matrices de confusión y matriz de confusión de segundo orden, para comprobar el funcionamiento de ambos clasificadores, y para presentar la metodología de evaluación.

Para estos ejemplos, se ha utilizado un HMM continuo con los siguientes valores:

$$\lambda = (\pi, A, \mu, \sigma) \tag{11.7}$$

$\pi = [1; 0]$
$A = [0{,}9952, 0{,}0048; 0{,}0081, 0{,}9919]$
$\mu = [0{,}8493, 1{,}8168]$
$\sigma(:, :, 1) = 0{,}0827$
$\sigma(:, :, 2) = 3{,}3582$

Mientras que el árbol utilizado es del tipo *classregtree*, pero por ser demasiado extenso, no mostramos sus parámetros en este documento, pero están disponibles en el código descrito en A en la página 130.

Capítulo 12

Resultados de un HMM frente a un árbol binario

Se presenta esta serie de cinco ejemplos, para mostrar el funcionamiento del clasificador basado en modelos ocultos de Markov, frente a un árbol binario. Se ha entrenado al modelo oculto de Markov con una única trayectoria (*trayectoria_22*), mientras que el árbol fue entrenado tras pruebas con varias trayectorias con la *trayectoria_14* por ser la trayectoria que mejores resultados parecía ofrecer. Dado que el ámbito de este proyecto está más centrado en desarrollar un modelo que sea capaz de clasificar de forma correcta las trayectorias y además de evaluar dichas clasificaciones, en vez de entrenar ambos modelos con la misma trayectoria, hemos buscado los conjuntos de entrenamiento que mejores resultados parecían dar en vez de buscar una metodología correcta de entrenamiento. En cualquier caso, los resultados de entrenar al HMM con la misma trayectoria que la indicada para el árbol daban resultados similares.

También debemos indicar que los resultados con una serie tan pequeña no pretenden ser indicativos del funcionamiento de ambos algoritmos, se utilizaron simplemente para presentar la herramienta de clasificación y la de evaluación.

12.1. Trayectoria 1

Para la primera prueba utilizaremos una trayectoria recta:

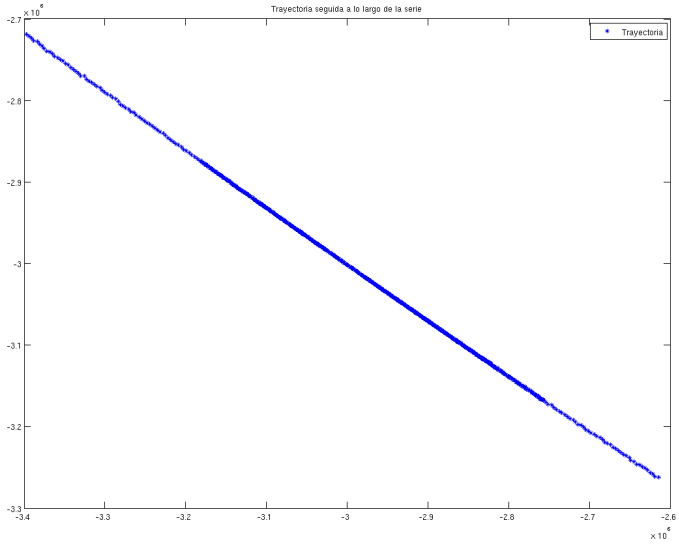


Figura 12.1: Trayectoria de prueba número 1

Esta trayectoria no tiene ninguna maniobra y sus residuos son:

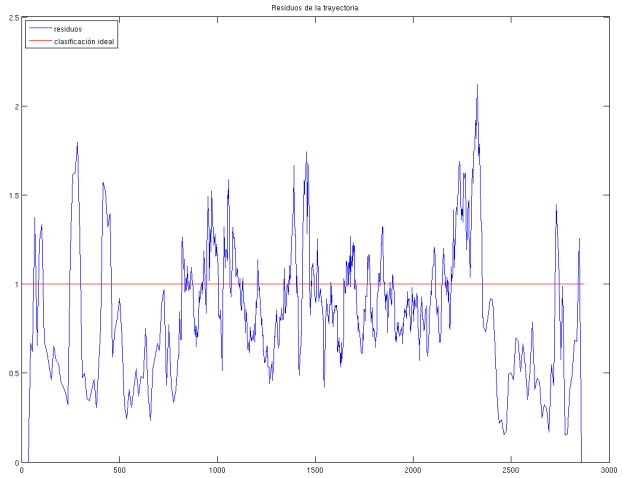


Figura 12.2: Trayectoria de prueba número 1

Los residuos presentan, un cierto nivel de ruido en ciertos puntos de la trayectoria, que pueden confundir a los algoritmos. En cualquier caso, su clasificación ideal es la mostrada por la línea roja, siendo '1' no maniobra, y los valores positivos un cambio de maniobra y los negativos la vuelta desde el estado de maniobra al de no maniobra.

Como se puede observar, la clasificación de dicha trayectoria por el HMM presentado 12.1 en la página 111en presenta dos fallos en zonas de mucho nivel de ruido:

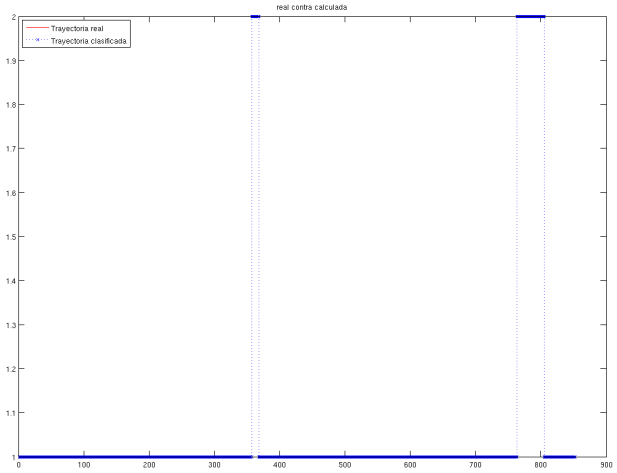


Figura 12.3: Trayectoria de prueba número 1

El árbol a su vez, también presenta algunos problemas debido a los puntos con más ruido de los residuos, a los que confunde con cambios de trayectoria:

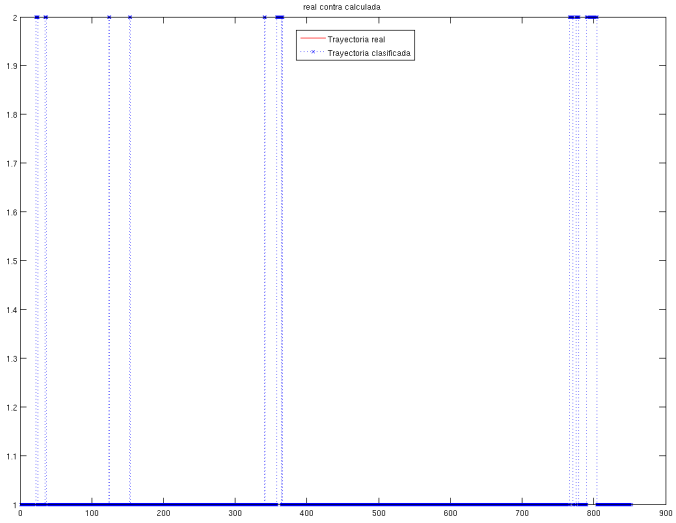


Figura 12.4: Trayectoria de prueba número 1

Los resultados del evaluador para ambas series temporales son, enfrentando

a las matrices clásicas de confusión frente a las de orden superior, para mostrar sus comportamientos y las ventajas de utilizar la matriz de orden superior, que aunque en este ejemplo son muy claros por que el HMM no ha tenido fallos, en otros ejemplos muestran mejor su potencial:

Matriz clásica	Matriz de confusión de orden superior
$\begin{bmatrix} 799 & 53 \\ 0 & 0 \end{bmatrix}$	$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 2 & 2 & 847 \end{bmatrix}$

Cuadro 12.1: Resultados de las matrices de confusión y de orden superior para el ejemplo 1 con el HMM

Matriz clásica	Matriz de confusión de orden superior
$\begin{bmatrix} 814 & 38 \\ 0 & 0 \end{bmatrix}$	$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 10 & 10 & 831 \end{bmatrix}$

Cuadro 12.2: Resultados de las matrices de confusión y de orden superior para el ejemplo 1 con el HMM

Con este último ejemplo se puede observar que la matriz de confusión señala un número aparentemente pequeño de errores, concretamente de un 4,46 %, lo que nos llevaría a considerarlo una buena clasificación (8.1.2 en la página 78). La matriz de orden superior se centra en analizar el número de transiciones clasificadas para analizar el comportamiento del algoritmo y no en los totales de estados clasificados correcta o incorrectamente, de forma que muestra un total de diez maniobras incorrectamente clasificadas y que al ser de pequeños tamaño indican una alta sensibilidad al ruido.

12.2. Trayectoria 2

La segunda maniobra clasificada es la mostrada a continuación:

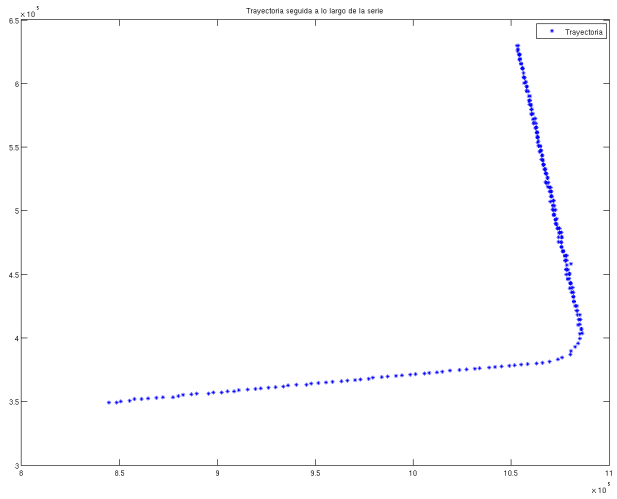


Figura 12.5: Trayectoria de prueba número 2

Esta trayectoria presenta una maniobra muy marcada y un nivel de ruido algo elevado, debido principalmente al escaso número de muestras que la forman:

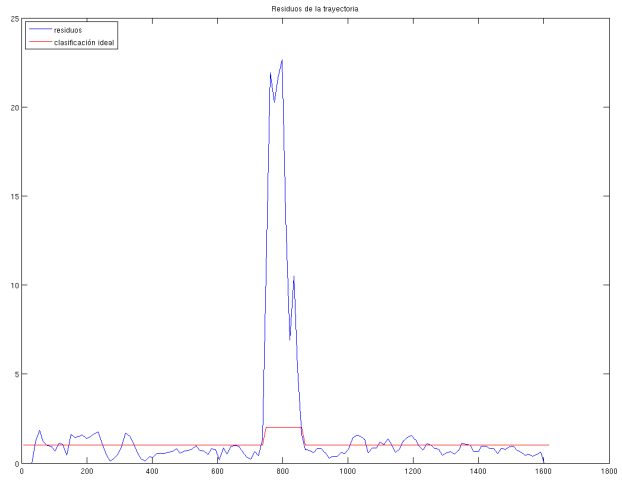


Figura 12.6: Trayectoria de prueba número 2

Aún teniendo en cuenta dicho ruido, el HMM parece ser capaz de clasificarla de forma correctamente, si bien se observa un fenómeno que tiende a repetirse con este modelo en particular: ante series en las cuales los primeros y últimos residuos presentan grandes variaciones, el modelo tiende a clasificar la primera y/o última zona como maniobra, cuando realmente se encuentra al final de la serie. Este pequeño problema es evitable, ampliando las tomas de muestras y

prescindiendo de los primeras y últimas 50 muestras a la hora de presentar las gráficas para no tener en cuenta dichos errores, especialmente marcados en determinados ejemplos, pero se ha decidido mostrar para comparar ambos algoritmos y considerarlo como error:

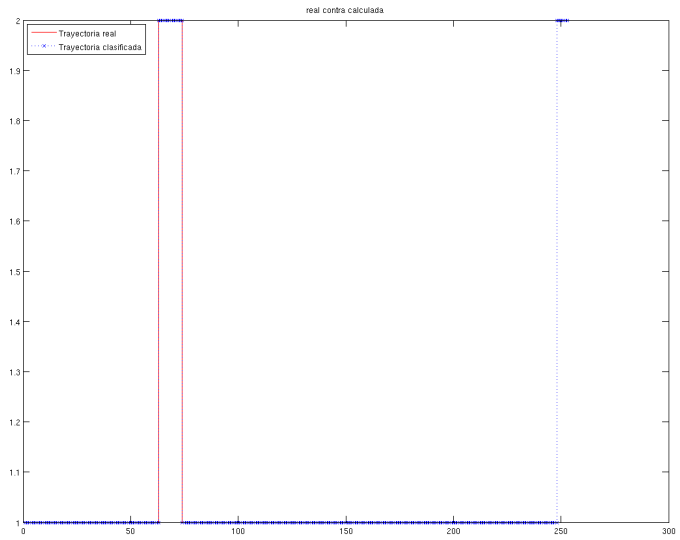


Figura 12.7: Trayectoria de prueba número 2

El árbol a diferencia del HMM, presenta muchos problemas con el ruido presente en la trayectoria, que le fuerzan a realizar muchas clasificaciones incorrectas de pequeño tamaño:

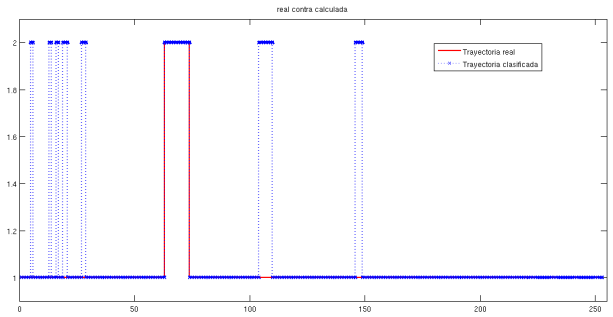


Figura 12.8: Trayectoria de prueba número 2

Los resultados del evaluador para ambas series temporales son, enfrentando a las matrices clásicas de confusión frente a las de orden superior, para mostrar sus comportamientos y las ventajas de utilizar la matriz de orden superior, que aunque en este ejemplo son muy claros por que el HMM no ha tenido fallos, en otros ejemplos muestran mejor su potencial:

Matriz clásica	Matriz de confusión de orden superior
$\begin{bmatrix} 236 & 6 \\ 0 & 11 \end{bmatrix}$	$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 1 & 0 & 249 \end{bmatrix}$

Cuadro 12.3: Resultados de las matrices de confusión y de orden superior para el ejemplo 2 con el HMM

Matriz clásica	Matriz de confusión de orden superior
$\begin{bmatrix} 226 & 16 \\ 0 & 11 \end{bmatrix}$	$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 7 & 7 & 236 \end{bmatrix}$

Cuadro 12.4: Resultados de las matrices de confusión y de orden superior para el ejemplo 2 con el HMM

De nuevo, se observa que la matriz de confusión indica relativamente pocos errores de clasificación, pero la matriz de orden superior está en cambio indicando que el clasificador aunque acierta en una transición, clasifica como transiciones muchas estados en los cuales no se maniobra.

12.3. Trayectoria 3

El tercer ejemplo muestra una aproximación a un aeropuerto, en la cual la aeronave realiza cuatro maniobras en forma de hipódromo para perder velocidad y altura antes de aterrizar. En este caso pues, se dan cuatro maniobras y bajo nivel de ruido:

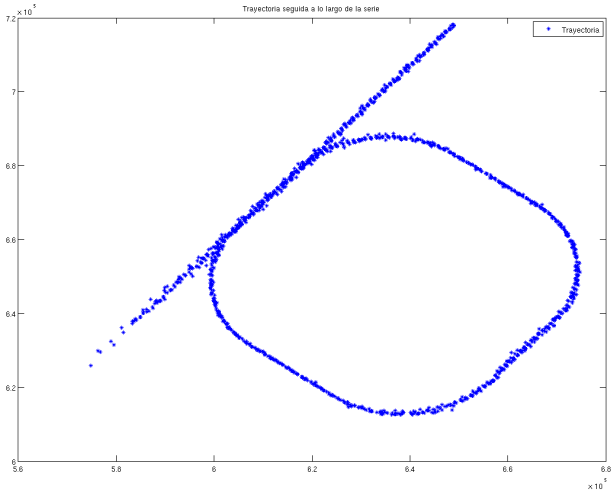


Figura 12.9: Trayectoria de prueba número 3

Esta trayectoria cuatro maniobras marcadas, las cuales se presentan a continuación:

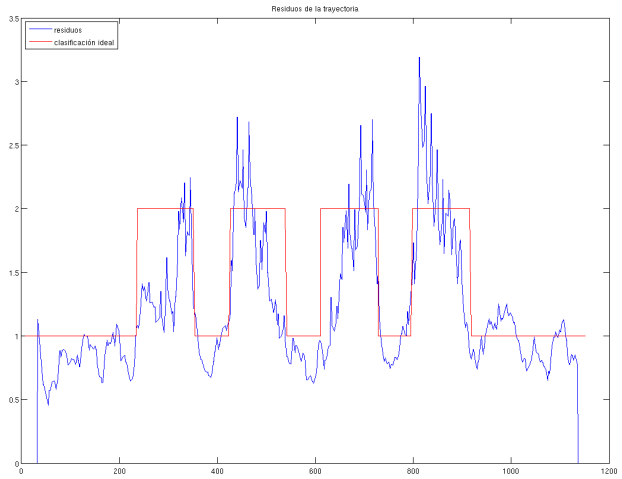


Figura 12.10: Trayectoria de prueba número 1

El HMM es capaz de clasificar de forma aceptable la trayectoria (si bien es cierto que con cierto retraso en alguna de las transiciones y detectando una transición de más por el problema explicado en 12.2 en la página 116):

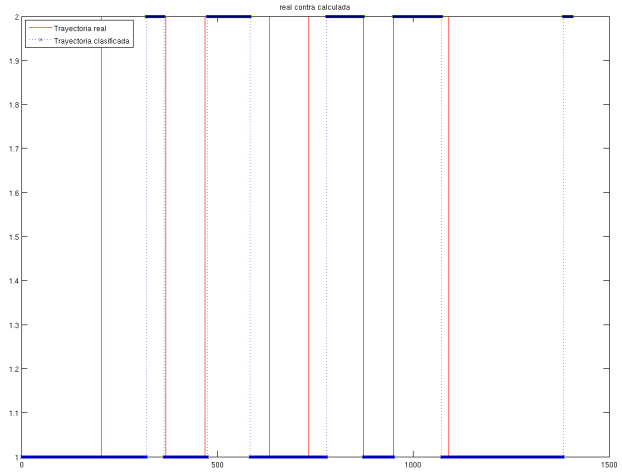


Figura 12.11: Trayectoria de prueba número 3

El árbol también detecta varias áreas de maniobra, pero su mayor sensibilidad le hacen detectar transiciones de más:

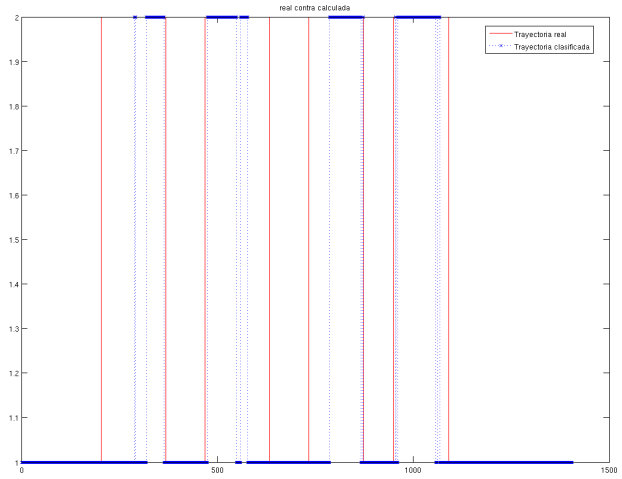


Figura 12.12: Trayectoria de prueba número 3

Esta maniobra es un ejemplo muy claro de las ventajas del uso de la matriz de orden superior, porque se muestra claramente que mientras que la matriz de confusión parece indicar claramente que el árbol es mejor que el HMM, la matriz de orden superior nos indica que las clasificaciones del HMM son más correctas que las del árbol:

Matriz clásica	Matriz de confusión de orden superior
$\begin{bmatrix} 773 & 25 \\ 238 & 370 \end{bmatrix}$	$\begin{bmatrix} 4 & 0 & 0 \\ 0 & 4 & 0 \\ 1 & 0 & 1394 \end{bmatrix}$

Cuadro 12.5: Resultados de las matrices de confusión y de orden superior para el ejemplo 3 con el HMM

Matriz clásica	Matriz de confusión de orden superior
$\begin{bmatrix} 797 & 1 \\ 279 & 329 \end{bmatrix}$	$\begin{bmatrix} 4 & 0 & 0 \\ 0 & 4 & 0 \\ 5 & 5 & 1387 \end{bmatrix}$

Cuadro 12.6: Resultados de las matrices de confusión y de orden superior para el ejemplo 3 con el HMM

Así, se muestra que el árbol tiene más resultados incorrectos que correctos, frente al aparente buen resultado del árbol comparado con el HMM, si se utiliza tan solo la matriz de confusión.

12.4. Trayectoria 4

Para el cuarto ejemplo utilizamos una trayectoria con mucho ruido, que presenta una única maniobra debido a un pequeño cambio de trayectoria (de hecho, difícil de observar a simple vista debido al ruido de la misma):

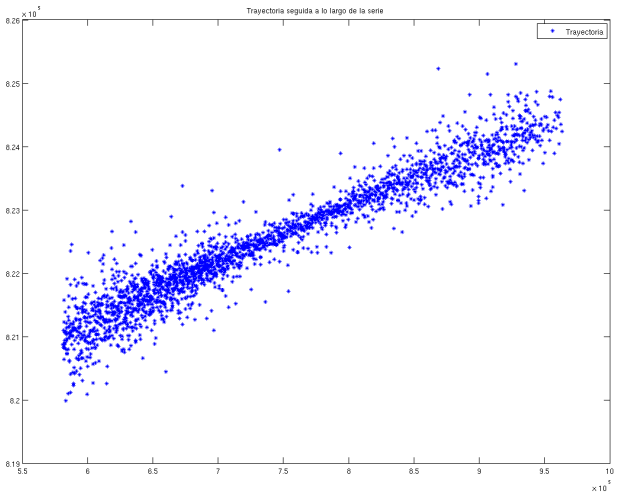


Figura 12.13: Trayectoria de prueba número 4

El cambio de trayectoria se muestra a continuación, observando de hecho que resulta muy complicado diferenciarlo, puesto que el instante en que está maniobrando, existen los mismos niveles de residuos que en otras partes de la trayectoria:

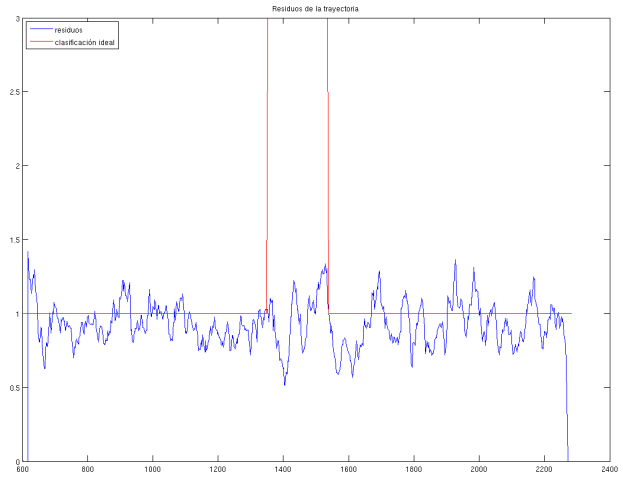


Figura 12.14: Trayectoria de prueba número 4

El HMM es incapaz de detectar el cambio de trayectoria y considera que es completamente recta, obviando los puntos inicial y final, que al pasar de niveles de residuo de '0' a '1' de forma casi instantánea, los considera como zonas de trayectoria:

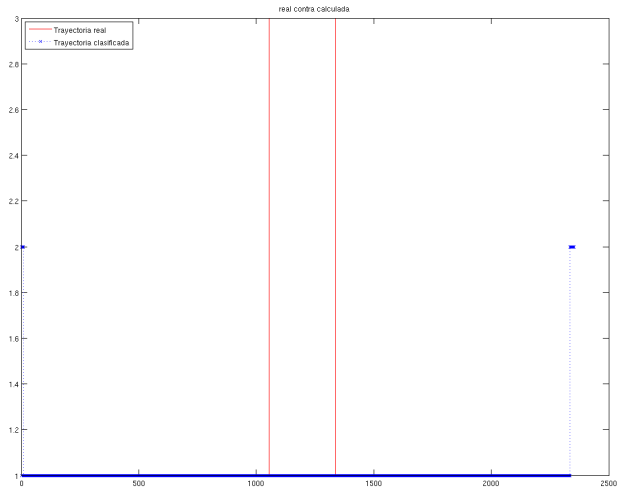


Figura 12.15: Trayectoria de prueba número 4

El árbol tampoco detecta cambio alguno de trayectoria:

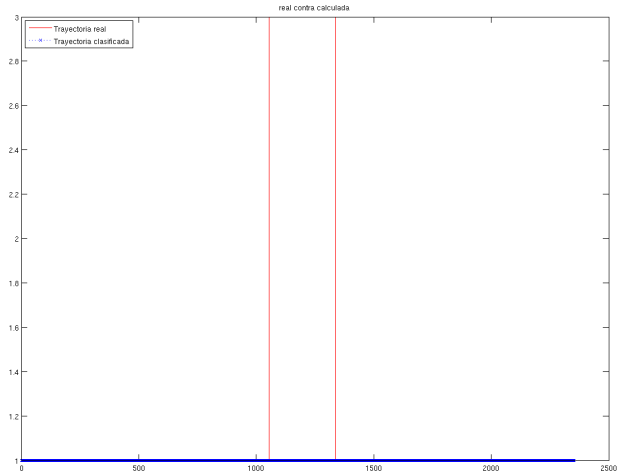


Figura 12.16: Trayectoria de prueba número 4

Los resultados de las matrices de confusión de ambos ejemplos son por tanto los mismos:

Matriz clásica	Matriz de confusión de orden superior
$\begin{bmatrix} 2244 & 26 \\ 282 & 0 \end{bmatrix}$	$\begin{bmatrix} 0 & 0 & 1 \\ 0 & 0 & 1 \\ 2 & 1 & 2346 \end{bmatrix}$

Cuadro 12.7: Resultados de las matrices de confusión y de orden superior para el ejemplo 4 con el HMM

Matriz clásica	Matriz de confusión de orden superior
$\begin{bmatrix} 2070 & 0 \\ 282 & 0 \end{bmatrix}$	$\begin{bmatrix} 0 & 0 & 1 \\ 0 & 0 & 1 \\ 0 & 0 & 2349 \end{bmatrix}$

Cuadro 12.8: Resultados de las matrices de confusión y de orden superior para el ejemplo 4 con el HMM

12.5. Trayectoria 5

El último ejemplo de esta serie de trayectorias es básicamente opuesto al anterior: se trata de una trayectoria recta con dos zonas inicial y final que presentan niveles altos de ruido:

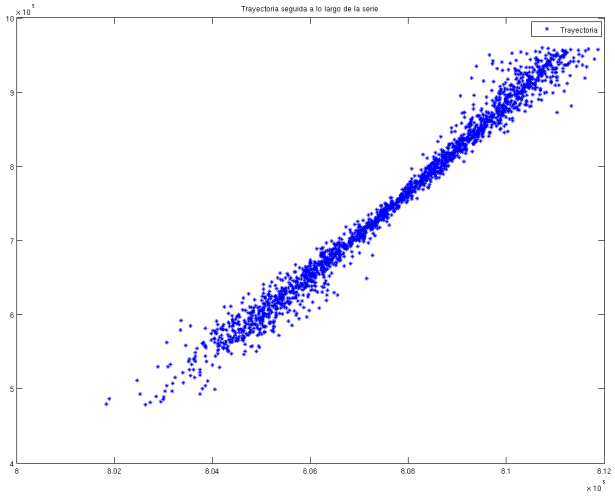


Figura 12.17: Trayectoria de prueba número 5

Vemos por tanto una trayectoria no demasiado compleja, sin cambios de dirección:

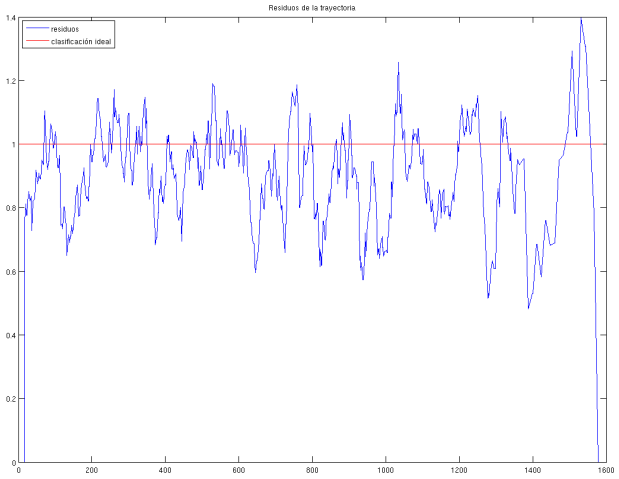


Figura 12.18: Trayectoria de prueba número 5

El HMM la clasifica de forma correcta (si se obvian las transiciones inicial y final):

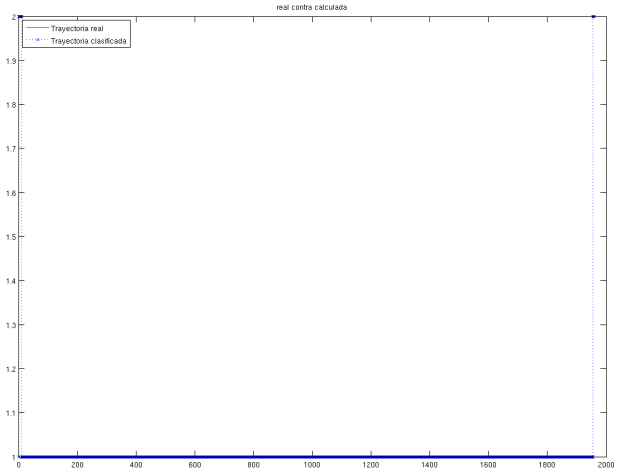


Figura 12.19: Trayectoria de prueba número 5

El árbol tampoco detecta cambio alguno de trayectoria, clasificándola correctamente:

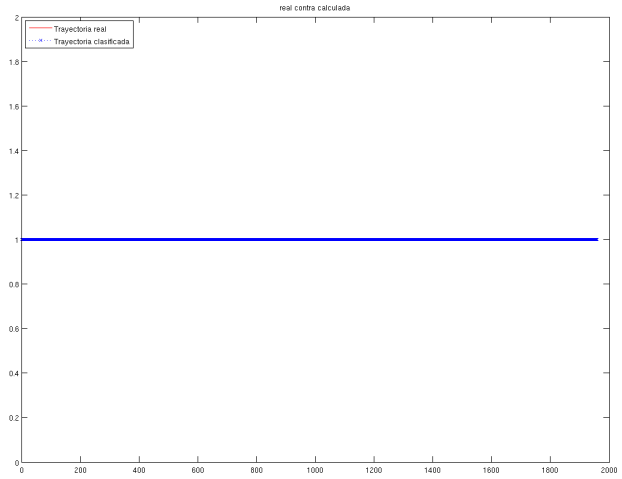


Figura 12.20: Trayectoria de prueba número 5

Los resultados de las matrices de confusión de ambos ejemplos son muy similares:

Matriz clásica	Matriz de confusión de orden superior
$\begin{bmatrix} 1944 & 14 \\ 0 & 0 \end{bmatrix}$	$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 2 & 1 & 1954 \end{bmatrix}$

Cuadro 12.9: Resultados de las matrices de confusión y de orden superior para el ejemplo 5 con el HMM

Matriz clásica	Matriz de confusión de orden superior
$\begin{bmatrix} 1958 & 0 \\ 0 & 0 \end{bmatrix}$	$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1957 \end{bmatrix}$

Cuadro 12.10: Resultados de las matrices de confusión y de orden superior para el ejemplo 5 con el HMM

12.6. Conclusiones de la primera tanda de trayectorias

El HMM elegido, mezcla de dos gaussianas de medias $\sigma_1 = 8,493$ y $\sigma_2 = 3,3582$, con unas probabilidades de emisión dadas por $\mu_1 = 0,8493$ y $\mu_2 = 1,8168$, un estado inicial $I = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$ y una matriz de probabilidades de transición $A = \begin{bmatrix} 0,9952 & 0,0048 \\ 0,0081 & 0,9919 \end{bmatrix}$ presenta mejor capacidad para clasificar de forma

correcta los cambios de trayectoria.

Además, dicho modelo ha sido entrenado con una sola trayectoria, mientras que el árbol requirió tres trayectorias para alcanzar los resultados mostrados. Es cierto que el HMM presenta algunos problemas, principalmente errores de clasificación al inicio y final de las trayectorias, pero al estar presentes en muchas ocasiones es posible obviarlos. Se han presentado, eso sí, a fin de indicar los problemas con que se encontraría una aplicación que utilizara dicho modelo.

Otro hecho a resaltar es que si en vez de entrenar el HMM con la trayectoria utilizada, se alcanza un modelo muy similar que ofrece los mismos resultados. Esto indica que ambos son subóptimos y que el HMM es, en principio, fácil de entrenar y ofrece, con poco entrenamiento muy buenos resultados.

También se han mostrado las ventajas del uso de una matriz de orden superior para llevar a cabo la evaluación de resultados de trayectorias.

Parte VI

Conclusiones y Futuros Desarrollos

Nos propusimos dos objetivos de este Proyecto de Fin de Carrera: el primero el estudio de una herramienta de clasificación de series temporales basada en modelos ocultos de Markov y el segundo diseñar una metodología de evaluación de clasificadores.

Respecto al primero de los objetivos, elegimos los modelos ocultos de Markov como herramienta clasificadora por considerar que, debido a sus características, tendría ventajas al reconocer patrones complejos frente a otros algoritmos más simples.

Fruto del estudio de dicho problema, hemos buscado diferentes implementaciones de modelos ocultos de Markov hasta dar con una (el *HMM Murphy Toolbox*) que integra toda la funcionalidad que buscábamos, y que hemos aplicado, creemos que con buenos resultados, a un problema concreto de clasificación de trayectorias, demostrando las fortalezas de dichos modelos.

Por supuesto, este tipo de modelos también tiene desventajas, algunas son debidas a su naturaleza y otras, como la de su evaluación, son debidas a que las metodologías habitualmente utilizadas no resultan prácticas para este tipo de modelos.

Cuando nos dimos cuenta de dicha problemática, decidimos abrir otra vía de investigación y desarrollar una metodología que tuviera en cuenta la problemática asociada a la evaluación de modelos ocultos de Markov, que acabó convirtiéndose en el segundo de los objetivos de este Proyecto de Fin de Carrera.

Dicha metodología ha sido diseñada teniendo en cuenta metodologías y herramientas ya probadas, cuya funcionalidad se ha adaptado para que sean capaces de evaluar correctamente no solo los modelos ocultos de Markov, si no también otros clasificadores, de forma que de lugar a una herramienta flexible.

Con todo esto, consideramos que hemos cumplido los objetivos de este Proyecto de Fin de Carrera. Durante el estudio de los problemas antes descritos y de las fuentes de documentación, hemos tenido que descartar algunas posibles vías de investigación, principalmente por estar fuera del ámbito del proyecto, aunque consideramos que son los bastante interesantes como para proponerlas como futuras posibilidades:

La primera tiene que ver con el entrenamiento de los modelos ocultos de Markov; en nuestro caso se han entrenado utilizando el algoritmo Baum-Welch [4], que como ya hemos comentado, presenta deficiencias al ser un algoritmo de búsqueda local. Por esta razón, proponemos la posibilidad de utilizarlo de forma conjunta con un algoritmo de búsqueda global. El algoritmo de *simmulated annealing* [16] parece haberse utilizado junto al Baum-Welch y podría ser una buena opción.

La segunda línea propuesta consistiría en el desarrollo de la metodología de evaluación propuesta y de la extensión de su funcionalidad: Por ejemplo, la matriz de orden superior que hemos desarrollado funciona de forma correcta con dos posibles estados, pero debería ser capaz de funcionar con más para ser de utilidad práctica con otros clasificadores o modelos ocultos de Markov más grandes.

Finalmente y como última vía de investigación relacionada con el problema del cálculo de trayectorias, hemos notado que en determinadas circunstancias se producen residuos de gran valor que los modelos ocultos de Markov no pueden tener en cuenta, debido a que, al estar basados en distribuciones normales, al encontrarse con residuos cuyos valores se disparan, no son clasificados correctamente.

Aunque este problema se soluciona si se incrementa el tamaño de la ventana para el cálculo de mínimos cuadrados, esto supone que el modelo sea incapaz de reconocer cambios suaves en la trayectoria, mientras que si se reduce la ventana, si es capaz de reconocer los cambios de trayectorias pronunciados, pero pierde algunos pequeños cambios de trayectoria.

Como dichos residuos dependen del tamaño de la ventana que se selecciona para el cálculo de los mínimos cuadrados, proponemos la posibilidad de buscar un nuevo desarrollo de la herramienta para obtener los mínimos cuadrados que no den lugar a residuos tan grandes que confundan al modelo oculto de Markov, posiblemente vía reducir o aumentar el tamaño de la ventana para el cálculo de los mínimos cuadrados que dan lugar a los residuos.

Parte VII

Apéndices

Apéndice A

Manual de Usuario de la Herramienta de Clasificación

Este manual sirve como guía de uso de la herramienta de clasificación de series temporales en base a modelos ocultos de Markov.

Para dichos modelos utilizaremos como notación posible la siguiente:

- Discretos:

$$\lambda = (\pi, A, B)$$

  - π: probabilidades iniciales.
  - A: matriz de probabilidades de transición.
  - B : matriz de probabilidades de observación.

- Continuos:

$$\lambda = (\pi,\, A,\, \mu_{jm},\, \sigma_{jm},\, c_{jm})$$

  - π: probabilidades iniciales.
  - A: matriz de probabilidades de transición.
  - μ_{jm} :medias de las gaussianas. M gaussianas, N estados, $1 \leq m \leq M$, $1 \leq j \leq N$.
  - σ_{jm}: covarianzas de las gaussianas. M gaussianas, N estados, $1 \leq m \leq M$, $1 \leq j \leq N$.
  - c_{jm}: pesos de las gaussianas (de no incluirse se sobreentenderá que todas las gaussianas tienen el mismo valor). M gaussianas, N estados, $1 \leq m \leq M$, $1 \leq j \leq N$.

A.1. Instalación y requisitos

Esta herramienta requiere una versión de Matlab igual o superior a la versión 7.6.0 (R2008a).

Para instalar la librería, basta con añadirla al *path* de Matlab con el comando:

addpath(genpath('ruta'))

siendo *ruta* el directorio donde se ha descomprimido la librería.

Los archivos de *Matlab* que detallaremos en este manual pertenecen a la herramienta del *HMM Murphy Toolbox*, contenida en el subdirectorio *HMM*, al subirectorio *Clasificador* y al subdirectorio *Ejemplos*.

A.2. Entrenamiento discreto

La función *dhmm_em* se utiliza para entrenar una modelo discreto con el algoritmo de EM (realmente utiliza el algoritmo Baum-Welch, que es una simplificación del EM [4]). Esta función recibe como parámetros un modelo oculto de Markov discreto, formado por un vector con las probabilidades iniciales del modelo, una matriz de probabilidades de transición, una matriz de probabilidades de observación y un conjunto de datos de entrenamiento e intenta entrenar el modelo que recibe como parámetro, para mejorar la capacidad del mismo para reconocer dichas secuencias.

Como resultado, produce un modelo mejorado y una matriz logarítmica de probabilidades, que se puede utilizar para comprobar si entre sucesivas iteraciones del algoritmo se mejora la probabilidad de observar una determinada cadena de entrenamiento.

El principal problema encontrado con este algoritmo es que por su propia definición [4] se trata de un algoritmo de búsqueda local, con cierta tendencia a devolver máximos locales como resultado, con lo cual debe tenerse en cuenta que los puntos desde los que se parte el entrenamiento son de gran importancia: de haber máximos locales el algoritmo se detendrá en dichos máximos.

A.2.1. Llamada a la Función de Entrenamiento

La llamada a la función de entrenamiento continuo es:

dhmm_em(data, prior, transmat, obsmat, varargin)

Siendo los parámetros especificados:

- *data*: matriz formada por dos vectores con los datos observados y clasificados (caso de haber un único vector). De haber varios, se presentan tantas *cell*, cada una con los vectores con los datos observados.
  - *data = [observados clasificados]*
- *prior*: π; probabilidades iniciales.
- *transmat*: A; matriz de probabilidades de transición.
- *obsmat*: B; matriz de probabilidades de observación.
- *varargin*: argumentos opcionales:
  - *'maxIter'*: número máximo de iteraciones

vg. llamadas a la función con un solo vector (especificamos todos los parámetros):

inicial = [1;0];
transiciones = [0.9950, 0.0050; 0.0006, 0.9934];
mu = [0.8590, 1.6242];
obsmat = [0.9, 0.1; 0.06, 0.94];

dhmm_em([[1 2 2 2 1 1 1...] [1 1 2 2 2 1 1 ...]], inicial, transmat, obsmat)

y con múltiples vectores de entrenamiento, todos ellos en vectores de cell:

dhmm_em([cellsObs cellsHid], prior, transmat, obsmat)

Se puede probar a entrenar vía un conjunto de datos de entrenamiento y un HMM inicial aleatorio, pero como ya se ha explicado, esto dará malos resultados: el algoritmo es de búsqueda local y depende mucho de los valores iniciales (el HMM recibido como parámetro).

A.2.2. Resultados de la Función de Entrenamiento

La llamada a la función:

[LL, prior, transmat, obsmat, nrIterations] = dhmm_em(data, prior, transmat, obsmat, varargin)

Devuelve como resultado los parámetros del HMM entrenado en base a dichas observaciones:

- *prior*: π; probabilidades iniciales.
- *transmat*: A; matriz de probabilidades de transición.
- *obsmat*: B; matriz de probabilidades de observación.
- *nrIterations*: numero de iteraciones que se han realizado para alcanzar el máximo.
- *LL*: logprobabilidad de reconocer la o las secuencias de entrenamiento.

Ejemplo: Llamada a la función de entrenamiento:

data = [[1 2 2 2 1 1 1...] [1 1 2 2 2 1 1 ...]]
inicial = [1;0];
transiciones = [0.9950, 0.0050; 0.0006, 0.9934];
obsmat = [0.9, 0.1; 0.06, 0.94];

dhmm_em([[1 2 2 2 1 1 1...] [1 1 2 2 2 1 1 ...]], inicial, transiciones, mu, sigma)

A.3. Entrenamiento continuo

El entrenamiento continuo es bastante similar al discreto, salvo que al tratarse de HMM continuos, en lugar de utilizarse una matriz de probabilidades de observación, se utilizan funciones gaussianas de densidad de probabilidad para representar las probabilidades de las variables ocultas.

Estas gaussianas vienen dadas por dos parámetros, la media (μ) y la varianza (σ), de forma que la llamada a la función de entrenamiento continuo, *mhmm_em*, se hace tomando como parámetros los datos de entrenamiento, las matrices de probabilidad inicial y de transición, y los valores con la media y varianza de la gaussiana.

Como resultado, devuelve un modelo mejorado y una matriz logarítmica de probabilidades, que se puede utilizar para comprobar si entre sucesivas iteraciones del algoritmo, mejora la probabilidad de observar una determinada cadena de entrenamiento.

Al igual que con su variante discreta, esta función está basada en el algoritmo EM simplificado (Baum-Welch), de forma que se trata de una función de búsqueda local, por lo que en caso de encontrar máximos locales al entrenar un HMM los devolverá como resultado válido. Es por tanto muy dependiente de los valores iniciales que reciba como parámetro. Además existe una patología agravada: es posible que en determinados puntos la varianza de unos datos al aplicar el HMM tienda a 0, dando como resultado una probabilidad infinita, hecho que de ocurrir, suele darse cerca del máximo global (esto es, el modelo más perfecto para esa secuencia dada).

En muchos casos se parte del hecho de que de por sí solo, el algoritmo EM no es capaz de encontrar dicho punto para evitar dicha patología. De hecho esta función no la tiene en cuenta y en las pruebas realizados no se ha dado el caso, aunque no puede descartarse que en futuras pruebas se encontrara.

Sin embargo, es necesario tener en cuenta que podría darse dicho caso y que se daría justamente en la mejor solución posible.

Al entrenar un HMM con este algoritmo, existen otros problemas que debemos tener en cuenta:

- Disminución de la probabilidad logarítmica durante el entrenamiento: este caso sugiere que uno de los componentes de la mezcla no ofrece suficientes datos, lo que indica que sería mejor probar con menos clases de datos o bien que el HMM recibido es poco representativo (habría que hacer una mejor inicialización).
- Matriz de varianza singular (determinante de σ sea 0 con distribuciones multivariadas): puede darse en mezclas multivariadas, se puede dar con una muestra de datos pequeña o con una inicialización incorrecta del modelo (concretamente, de las medias μ). Puede arreglarse si se usa una mejor inicialización o se especifica como parámetro que la covarianza sea diagonal o esférica. Dado que los modelos utilizados en este proyecto no eran multivariados, no se ha observado este problema.

El principal problema encontrado con este algoritmo es que por su propia definición (7.5.2 en la página 68) se trata de un algoritmo de búsqueda local, con cierta tendencia a devolver máximos locales como resultado, con lo cual debe tenerse en cuenta que los puntos desde los que se parte el entrenamiento son de gran importancia.

A.3.1. Llamada a la Función de Entrenamiento

La llamada a la función de entrenamiento continuo recibe los parámetros de entrenamiento para calcular un HMM es:

gausshmm_train_observed(obsData, hiddenData, nstates, varargin)

Siendo los parámetros especificados:

- *obsData*: vector con los datos observados caso de haber un único vector. De haber varios, se presentan tantas *cell*, cada una con los vectores con los datos observados.
- *hiddenData*: vector con los datos clasificados reales correspondientes a los observados o vector de *cell,* cada una con los vectores con los datos clasificados.
- *nstates*: número de estados
- *varargin*: argumentos opcionales, pudiendo tomarse como valores los siguientes (uno o varios pueden estar presentes):
  - *'dirichletPriorWeight'*: suaviza los pesos de la matriz de transición
  - *'cov_type'*: *'full'*, *'diag'* o *'spherical'*: utilizado cuando se da una matriz de varianza singular en mezclas multivariadas. Por defecto es *'full'*.

vg. llamadas a la función con un solo vector:

gausshmm_train_observed([1 2 2 2 1 1 1...], [1 1 2 2 2 1 1 ...], 2)

y con múltiples vectores de entrenamiento, todos ellos en vectores de cell:

gausshmm_train_observed(cellsObs, cellsHid, 2)

A.3.2. Resultados de la Función de Entrenamiento

La llamada a la función:

[inicial, transmat, mu, sigma] = gausshmm_train_observed(obsData, hiddenData, nstates, varargin)

Devuelve como resultado los parámetros del HMM entrenado en base a dichas observaciones:

- *inicial*: π; probabilidades iniciales.
- *transmat*: A; matriz de probabilidades de transición.
- *mu*; μ_{jm};medias de las gaussianas.
- *sigma*:σ_{jm}; covarianzas de las gaussianas.

Se obtiene con esta función unos buenos parámetros de entrenamiento, partiendo de una o varias secuencias de observaciones.

Ejemplo: Llamada a la función de entrenamiento continuo (suponemos mezcla de dos gaussianas, podrían ser más):

gausshmm_train_observed([1 2 2 2 1 1 1...] [1 1 2 2 2 1 1 ...], 2)

A.3.3. Llamada a la Función de Entrenamiento Iterativo

A utilizar si se dispone de un HMM inicial o se conoce un modelo del que partir, la llamada a la función de entrenamiento continuo iterativa (Baum-Welch) es:

mhmm_em(data, prior, transmat, mu, sigma, varargin)

Siendo los parámetros especificados:

- *data*: matriz formada por dos vectores con los datos observados y clasificados (caso de haber un único vector). De haber varios, se presentan tantas *cell*, cada una con los vectores con los datos observados.
  - *data = [observados clasificados]*
- *prior*: π; probabilidades iniciales.
- *transmat*: A; matriz de probabilidades de transición.
- *mu*; μ_{jm};medias de las gaussianas.
- *sigma*:σ_{jm}; covarianzas de las gaussianas.
- *varargin*: argumentos opcionales:
  - *'maxIter'*: número máximo de iteraciones

vg. llamadas a la función con un solo vector:

mhmm_em([[1 2 2 2 1 1 1...] [1 1 2 2 2 1 1 ...]], prior, transmat, obsmat)

y con múltiples vectores de entrenamiento, todos ellos en vectores de cell:

mhmm_em([cellsObs cellsHid], prior, transmat, obsmat)

Se puede probar a entrenar vía un conjunto de datos de entrenamiento y un HMM inicial aleatorio, pero como ya se ha explicado, esto dará malos resultados: el algoritmo es de búsqueda local y depende mucho de los valores iniciales (el HMM recibido como parámetro).

Ejemplo: Llamada a la función de entrenamiento:
data = [[1 2 2 2 1 1 1...] [1 1 2 2 2 1 1 ...]]
inicial = [1;0];
transiciones = [0.9950, 0.0050; 0.0006, 0.9934];
sigma(:,:,1)=0.0601;
sigma(:,:,2)=0.2383;

mhmm_em([[1 2 2 2 1 1 1...] [1 1 2 2 2 1 1 ...]], inicial, transiciones, mu, sigma)

A.3.4. Resultados de la Función de Entrenamiento Iterativo

La llamada a la función:

[LL, prior, transmat, obsmat, nrIterations] = mhmm_em(data, prior, transmat, obsmat, varargin)

Devuelve como resultado los parámetros del HMM entrenado en base a dichas observaciones:

- *prior*: π; probabilidades iniciales.
- *transmat*: A; matriz de probabilidades de transición.
- *mu*; μ_{jm};medias de las gaussianas.
- *sigma*:σ_{jm}; covarianzas de las gaussianas.
- *nrIterations*: numero de iteraciones que se han realizado para alcanzar el máximo.
- *LL*: logprobabilidad de reconocer la o las secuencias de entrenamiento.

Ejemplo: Llamada a la función de entrenamiento iterativo discreto:
data = [[1 2 2 2 1 1 1...] [1 1 2 2 2 1 1 ...]]
inicial = [1;0];
transiciones = [0.9950, 0.0050; 0.0006, 0.9934];
mu = [0.8590, 1.6242];
sigma(:,:,1)=0.0601;
sigma(:,:,2)=0.2383;

mhmm_em([[1 2 2 2 1 1 1...] [1 1 2 2 2 1 1 ...]], inicial, transiciones, mu, sigma)

A.4. Clasificación de secuencias dado un HMM discreto

La función *dhmm_logprob* se puede utilizar para clasificar una secuencia dado un modelo, devolviendo como resultado la probabilidad logarítmica de que en un HMM se de un determinado conjunto de observaciones.

Se utiliza cuando se tienen varios HMM candidatos a generar una determinada secuencia y se quiere saber cual de todos ellos es el que tiene más probabilidades de reconocerla.

También resulta de utilidad para entrenamientos y evaluación de modelos ocultos de Markov por que genera un coeficiente, dados un modelo y un conjunto de observaciones, de forma que permite dibujar una curva de aprendizaje para una secuencia dada y sucesivos entrenamientos de un modelo.

Recibe como parámetros una secuencia de observaciones, y las matrices de probabilidades inicial, de transición y de observación, devolviendo la probabilidad de que el modelo dado haya generado dicha secuencia de observaciones.

Los datos se asume que pertenecen a un alfabeto $\{1, 2 \dots S\}$, siendo S el tamaño de la matriz de transición recibido. No pueden contener '0'.

A.4.1. Entradas del Clasificador Discreto

La llamada a la función tiene las siguientes entradas:

[loglik, errors] = dhmm_ logprob(data, prior, transmat, obsmat)

- *data*: observaciones, O
- *prior*: π; probabilidades iniciales.
- *transmat*: A; matriz de probabilidades de transición.
- *obsmat*: B; matriz de probabilidades de observación.

A.4.2. Salidas del Clasificador Discreto

La llamada a la función :

[loglik, errors] = dhmm_ logprob(data, prior, transmat, obsmat)

devuelve los siguientes datos:

- *loglik*: log probabilidad de observar la o las secuencias de datos.
- *errors*: errores en las secuencias

Ejemplo: Llamada al clasificador discreto:
inicial = [1;0];
transiciones = [0.9950, 0.0050; 0.0006, 0.9934];
obsmat = [0.9, 0.1; 0.06, 0.94];

dhmm_ logprob([[1 2 2 2 1 1 1...] [1 1 2 2 2 1 1 ...]], inicial, transiciones, obsmat)

A.5. Clasificación de secuencias dado un HMM continuo

La función *mhmm_ logprob* es muy similar a *dhmm_ logprob*: se puede utilizar para clasificar una secuencia dado un modelo, devolviendo como resultado la probabilidad logarítmica de que en un HMM se dé un determinado conjunto de observaciones.

Al igual que su variante discreta, se puede utilizar cuando se tienen varios HMM candidatos a generar una determinada secuencia y se quiere saber cual de todos ellos es el que tiene más probabilidades de reconocerla, o para entrenamientos y evaluación de modelos ocultos de Markov, al generar un coeficiente dados un modelo y un conjunto de observaciones, permitiendo dibujar una curva de aprendizaje para una secuencia dada y sucesivos entrenamientos de un modelo.

Si bien es muy útil para evaluar HMM de esta manera, al no tener en cuenta toda la funcionalidad que era necesaria para evaluación de clasificadores descrita en 8 en la página 76, se optó por desarrollar una metodología de evaluación,

descrita en 9. En cualquier caso, esta función se puede utilizar para el entrenamiento de HMM y de hecho es utilizada por las funciones de entrenamiento discreto.

Recibe como parámetros una secuencia de observaciones, y las matrices de probabilidades inicial y de transición, así como los valores que definen las gaussianas de emisión de probabilidades ocultas, μ y σ, las medias y varianzas. Devuelve como resultado la probabilidad de que el modelo dado haya generado dicha secuencia de observaciones.

Los datos se asume que pertenecen a un alfabeto $\{1, 2 \dots S\}$, siendo S el tamaño de la matriz de transición recibido. No pueden contener '0'.

A.5.1. Entradas del Clasificador Continuo

La llamada a la función tiene las siguientes entradas:

[loglik, errors] = mhmm_ logprob(data, prior, transmat, mu, sigma)

- *data*: observaciones, O
- *prior*: π; probabilidades iniciales.
- *transmat*: A; matriz de probabilidades de transición.
- *mu*; μ_{jm};medias de las gaussianas.
- *sigma*:σ_{jm}; covarianzas de las gaussianas.

A.5.2. Salidas del Clasificador Continuo

La llamada a la función :

[loglik, errors] = mhmm_ logprob(data, prior, transmat, mu, sigma)

devuelve los siguientes datos:

- *loglik*: log probabilidad de observar la o las secuencias de datos.
- *errors*: errores en las secuencias

Ejemplo: Llamada a la función de entrenamiento iterativo discreto:
data = [[1 2 2 2 1 1 1...] [1 1 2 2 2 1 1 ...]]
inicial = [1;0];
transiciones = [0.9950, 0.0050; 0.0006, 0.9934];
mu = [0.8590, 1.6242];
sigma(:,:,1)=0.0601;
sigma(:,:,2)=0.2383;

mhmm_ logprob([[1 2 2 2 1 1 1...] [1 1 2 2 2 1 1 ...]], inicial, transiciones, mu, sigma)

A.6. Generar una Secuencia de Observaciones de Prueba dado un HMM Discreto

Se puede obtener una secuencia de observaciones que incluya datos observados y las transiciones ocultas mediante la función dhmm_sample, que recibe como parámetros las matrices que definen un HMM discreto así como las longitudes de la secuencia y el número de secuencias a generar basados en estos parámetros. Este último puede ser necesario porque en determinados casos son necesarias múltiples secuencias de observaciones.

Se puede utilizar para generar secuencias artificiales de pruebas que incluyen además de la observación las transiciones ocultas, de forma que facilite los entrenamientos o las evaluaciones de los HMM.

Recibe como parámetros las longitudes de secuencia y número de secuencias a generar, cuando haga falta generar múltiples secuencias de observación, así como las matrices que definen el modelo de Markov: probabilidades inicial, de transición y de observación.

Como resultado, genera una secuencia de valores observados y una secuencia de los valores ocultos que han dado lugar a dichas observaciones.

A.6.1. Entradas del Generador de Observaciones

La llamada a la función tiene las siguientes entradas:

[obs, hidden] = dhmm_sample(initial_prob, transmat, obsmat, numex, len)

- *initial_prob*: π; probabilidades iniciales.
- *transmat*: A; matriz de probabilidades de transición.
- *obsmat*: B; matriz de probabilidades de observación.
- *numex*: número de secuencias a generar
- *len*: longitud de las secuencias a generar

A.6.2. Salidas del Generador de Observaciones

La llamada a la función :

[obs, hidden] = dhmm_sample(initial_prob, transmat, obsmat, numex, len)

devuelve los siguientes datos:

- *obs*: datos observador.
- *hidden*: datos reales.

Los resultados tendrán *numex* filas y *len* columnas.

Ejemplo: Llamada al generador de observaciones la función de entrenamiento:
inicial = [1;0];
transiciones = [0.9950, 0.0050; 0.0006, 0.9934];
obsmat = [0.9, 0.1; 0.06, 0.94];

dhmm_sample(inicial, transiciones, obsmat, 1, 1000)

A.7. Generar una secuencia de observaciones de prueba dado un HMM continuo

Al igual que con el modelo discreto, se puede obtener una secuencia de observaciones que incluya datos observados y las transiciones ocultas mediante la función mhmm_sample.

Se puede utilizar para generar secuencias artificiales de pruebas que incluyen además de la observación las transiciones ocultas, de forma que facilite los entrenamientos o las evaluaciones de los HMM.

Recibe como parámetros las longitudes de secuencia y número de secuencias a generar (cuando haga falta generar múltiples secuencias de observación), así como las matrices probabilidades inicial y de transición y los valores que definen las gaussianas de emisión de probabilidades ocultas, μ y σ, las medias y varianzas.

Como resultado, genera una secuencia de valores observados y una secuencia de los valores ocultos que han dado lugar a dichas observaciones.

A.7.1. Entradas del Generador de Observaciones

La llamada a la función tiene las siguientes entradas:

[obs, hidden] = mhmm_sample(initial_prob, transmat, mu, sigma, numex, len)

- *initial_prob*: π; probabilidades iniciales.
- *transmat*: A; matriz de probabilidades de transición.
- *mu*; μ_{jm};medias de las gaussianas.
- *sigma*:σ_{jm}; covarianzas de las gaussianas.
- *numex*: número de secuencias a generar
- *len*: longitud de las secuencias a generar

A.7.2. Salidas del Generador de Observaciones

La llamada a la función :

[obs, hidden] = mhmm_sample(initial_prob, transmat, obsmat, numex, len)

devuelve los siguientes datos:

- *obs*: datos observador.
- *hidden*: datos reales.

Los resultados tendrán *numex* filas y *len* columnas.

Ejemplo: Llamada a la función del generador de observaciones continuo:
data = [[1 2 2 2 1 1 1...] [1 1 2 2 2 1 1 ...]]
inicial = [1;0];
transiciones = [0.9950, 0.0050; 0.0006, 0.9934];
mu = [0.8590, 1.6242];
sigma(:,:,1)=0.0601;
sigma(:,:,2)=0.2383;

mhmm_sample([[1 2 2 2 1 1 1...] [1 1 2 2 2 1 1 ...]], inicial, transiciones, mu, sigma)

A.8. Cálculo de la secuencia oculta más probable

La función *viterbi* se utiliza para llamar a la función *viterbi_path*, teniendo dos posibles llamadas:

- Discreta: recibe como parámetros las matrices de probabilidad inicial (π), de transición (A), de observación (B) y la secuencia de datos de observación (O), en formato vector o matriz.
- Continua, similar a la anterior, pero como entrada: probabilidades iniciales (π), probabilidades de transición (A), media de las gaussianas (μ), varianza de las gaussianas (σ) y datos observados (O), en formato vector o matriz. También se puede especificar un vector de ponderación (c), de no especificarse, se da el mismo valor de ponderación a todas las gaussianas a mezclar.

Para simplificar las llamadas se han desarrollado funciones que las realizan de forma automática en base a si son o no continuas.

A.8.1. Llamada a la Función Viterbi

La llamada a la función:

path=viterbi(prior, transmat, obsmat, data, varargin)

Toma valores distintos en base a si el modelo es continuo o discreto.

A.8.1.1. Modelo Discreto

Calcula la secuencia de estados más probable que puede haber generado una observación,

path=viterbi(prior, transmat, obsmat, data)

- *data*: observaciones, O
- *prior*: π; probabilidades iniciales.
- *transmat*: A; matriz de probabilidades de transición.
- *obsmat*: B; matriz de probabilidades de observación.

Como resultado, se obtiene:

- *path*: secuencia de observaciones más probable.

Ejemplo: Llamada a la función de entrenamiento iterativo discreto:
data = [1 2 2 2 1 1 1...]
inicial = [1;0];
transiciones = [0.9950, 0.0050; 0.0006, 0.9934];
obsmat = [0.9, 0.1; 0.06, 0.94];

viterbi(inicial, transiciones, obsmat, [1 2 2 2 1 1 1...])

A.8.1.2. Modelo Continuo

Calcula la secuencia de estados más probable que puede haber generado una observación,

path=viterbi(prior, transmat, mu, sigma, data)

- *data*: observaciones, O
- *prior*: π; probabilidades iniciales.
- *transmat*: A; matriz de probabilidades de transición.
- *mu*; μ_{jm};medias de las gaussianas.
- *sigma*:σ_{jm}; covarianzas de las gaussianas.
- *varargin*: vector opcional con los pesos de las gaussianas. De no introducirse, se toman los mismos pesos para todas las gaussianas.

Como resultado, se obtiene:

- *path*: secuencia de observaciones más probable.

Ejemplo: Llamada a la función de entrenamiento iterativo discreto:
data = [1 2 2 2 1 1 1...]
inicial = [1;0];
transiciones = [0.9950, 0.0050; 0.0006, 0.9934];
mu = [0.8590, 1.6242];
sigma(:,:,1)=0.0601;
sigma(:,:,2)=0.2383;

viterbi(inicial, transiciones, mu, sigma, [1 2 2 2 1 1 1...])

A.9. Funciones de Utilidad

Miscelánea con varias funciones de ayuda y ejemplo:

A.9.1. Carga de Ficheros

Existe una función que permite la carga de los ficheros de trayectorias contenidos en el subdirectorio *archivosm_rar* para simplificar las pruebas y ejemplos, la llamada es:

[trayectoria, real, tiempo] = cargaFichero(fichero)

Siendo su entrada:

- *fichero*: ruta del fichero.

Y sus salidas:

- *trayectoria*: datos de la trayectoria (residuo ponderado).
- *real*: clasificación real.
- *fichero*: instantes de tiempos (útil para mostrar gráficas).

A.9.2. Entrenamiento

Función que permite la carga de los ficheros de trayectorias contenidos en el subdirectorio *archivosm_rar* y su uso para entrenar un HMM continuo que reconozca las trayectorias:

[inicial, transmat, mu, sigma] = entrena_tray(ficheros)

Siendo su entrada:

- *ficheros*: ruta del fichero o ficheros (se debe enviar un vector de rutas de ficheros).

Y sus salidas dan lugar a un HMM continuo:

- *inicial*: probabilidades iniciales.
- *transmat*: probabilidades de transición.
- *mu*: medias de las gaussianas de densidad de observación.
- *sigma*: covarianza de las gaussianas de densidad de observación.

A.9.3. Funciones de Árboles

Permiten entrenar y utilizar árboles de clasificación, se han utilizado para ejemplos de prueba y como grupo de control del clasificador.

A.9.3.1. Entrenamiento de Árbol

Permite entrenar un árbol de clasificación:

[arbol] = arbol_tray(fichero, estados)

Parámetros:

- *fichero*: ruta del fichero de entrenamiento.
- *estados*: estados que puede tomar, deben ser no numéricos (debido a limitaciones del algoritmo), vg. *estados=['n', 'm']*.

Devuelve un árbol entrenado.

A.9.3.2. Cálculo de Trayectoria

Permite utilizar un árbol de clasificación para clasificar una trayectoria:

[tray] = arbol_ calcula_ tray(residuos, tree, estados)

Parámetros:

- *residuos*: rediduos de la trayectoria.
- *tree*: árbol de clasificación.
- *estados*: estados que puede tomar, deben ser no numéricos (debido a limitaciones del algoritmo), vg. *estados=['n', 'm']*.

Devuelve una clasificación de la trayectoria.

A.10. Función score

Automatización de pruebas de evaluación de clasificadores y trayectorias para HMM y árboles. Situado en el subdirectorio *Ejemplos*.

Ejecuta una batería de tantas clasificaciones como ficheros reciba para un HMM por defecto o especificado; o para un árbol de clasificación.

[maC, confusion] = score(datos, tipo, varargin)

Siendo los posibles parámetros:

- *datos*: ficheros de datos con las trayectorias (array de *strings*).
- tipo $\begin{cases} 1 & HMM \\ 2 & Árbol \end{cases}$
  - Si es un HMM, se pueden especificar los valores de un HMM (π, A, μ y σ). De no especificarse, se utilizan los valores por defecto.
  - De querer ejecutar con un árbol, ha de introducirse un valor en el archivo de entrenamiento del árbol (llamado *ficheroARbol*, línea 36 del fichero *score.m*).

Salidas:

- *maC*: matriz de confusión (clásica).
- *confusion*: matriz de confusión de orden superior .

Ejemplo: Para árbol,

score(['ruta1', 'ruta2'], 2)

HMM por defecto,

score(['ruta1', 'ruta2'], 1)

HMM dado,
inicial = [1;0];
transiciones = [0.9950, 0.0050; 0.0006, 0.9934];
mu = [0.8590, 1.6242];
sigma(:,:,1)=0.0601;
sigma(:,:,2)=0.2383;

score(['ruta1', 'ruta2'], inicial, transiciones, mu, sigma)

A.11. Ejemplos

Además de todas las funciones indicadas en el manual mostrados, exisen ficheros con funciones de ejemplo ejecutables sin parámetro, que realizan clasificaciones y evaluaciones (situados en el subdirectorio *Ejemplos*):

- *hmm_ tiempo.m*: pequeño ejemplo de un HMM.
- *hmm_ bolas.m*: pequeño ejemplo de un HMM y clasificación por matriz de confusión clásica.
- *hmm_ casino.m*: ejemplo de funcionalidad utilizado para análisis de funcionalidad, sólo se ejecuta correctamente si se dispone del *Statistical Toolbox*.
- *hmm_ casino2.m*: similar al anterior.
- *hmm_ casino3.m*: similar al anterior, incorpora funcionalidad de la matriz de órden superior.

Apéndice B

Manual de Usuario de la Herramienta de Evaluación de Clasificadores

Este manual sirve como guía de uso de la herramienta de evaluación de clasificadores. Se ha pretendido que esta herramienta sea de uso no solo frente a clasificadores basados en modelos ocultos de Markov si no frente a otros clasificadores sin memoria más habituales, por lo cual es posible utilizarla para comprobar el rendimiento de varios clasificadores.

La metodología de evaluación usaremos una metodología implementada tiene en cuenta los siguientes puntos:

- Análisis de las transiciones entre estados.
- Comparar si los modelos son capaces de clasificar correctamente dichos cambios de estados.
- Obtención como resultado final de una matriz de confusión de orden superior.

B.1. Instalación y requisitos

Esta herramienta requiere una versión de Matlab igual o superior a la versión 7.6.0 (R2008a).

Para instalar la librería, basta con añadirla al *path* de Matlab con el comando:

addpath(genpath('ruta'))

siendo *ruta* el directorio donde se ha descomprimido la librería.

Los archivos de *Matlab* que detallaremos en este manual pertenecen a la herramienta del *HMM Murphy Toolbox*, contenida en el subdirectorio *HMM*, al subirectorio *Evaluador* y al subdirectorio *Ejemplos*.

B.2. Matriz de Confusión

La función *matriz_ confusión* genera una matriz de confusión, a partir de una secuencia calculada por un clasificador y de la secuencia real que debería haberse calculado. A partir de dicha matriz de tipo:

		Predecidos	
		Positivo	Negativo
Reales	Positivo	a	b
	Negativo	c	d

Cuadro B.1: Matriz de confusión

Es posible calcular los siguientes términos:

- Exactitud (AC), proporción del total de predicciones correctas:

$$AC = \frac{a+d}{a+b+c+d} \tag{B.1}$$

- Tasa de verdaderos positivos (TP), proporocion de positivos correctamente identificados:

$$TP = \frac{a}{a+b} \tag{B.2}$$

- Tasa de falsos positivos (FP), proporción de negativos incorrectamente clasificados como positivos:

$$FP = \frac{c}{c+d} \tag{B.3}$$

- Tasa de verdaderos negativos (TN), proporción de verdaderos negativos correctamente identificados:

$$TN = \frac{d}{c+d} \tag{B.4}$$

- Tasas de falsos negativos (FN), proporción de positivos incorrectamente clasificados como negativos:

$$FN = \frac{b}{a+b} \tag{B.5}$$

- La precisión (P), que es la proporción de positivos correctamente predichos:

$$P = \frac{a}{a+c} \tag{B.6}$$

La función *matriz_ confusión* genera a partir de las secuencias de datos real y calculada, y de los estados que existen (en los casos probados en este proyecto, se utiliza [1 2] para los estados 'estable' y 'maniobra', respectivamente) genera la matriz de confusión. Es posible calcular los términos indicados previamente con la función *terminos_ matriz_ confusion.*

B.2.1. Entradas de la Función de Cálculo de Matrices de Confusión

La función presenta la siguiente llamada:

[confusion, porcentaje] = matriz_ confusion(real, calculado, estados)

Dónde los parámetros especificados son:

- real: vector de datos correctamente clasificados.
- calculado: vector de datos calculados por el clasificador.
- estados: posibles estados que se pueden tomar.

Soporta cualquier tamaño de matriz de confusión.

B.2.2. Salidas de la Función de Cálculo de Matrices de Confusión

La función:

[confusion, porcentaje] = matriz_ confusion(real, calculado, estados)

Obtiene los resultados siguientes:

- *confusion*: matriz de confusion resultante.
- *calculado*: porcentajes de cada elemento de la matriz de confusion.

Ejemplo: Ejemplo de llamada a la función de cálculo de matrices de confusión:
calculado = [1 2 2 2 1 1 1...]
real= [1 1 2 2 2 1 1 ...]
estados=[1 2]

matriz_ confusion(real, calculado, estados)

B.3. Matriz de Confusión de Orden Superior

La función *matriz_ confusion_ orden2* calcula la matriz de confusión de orden superior i. Esta matriz esta compuesta de:

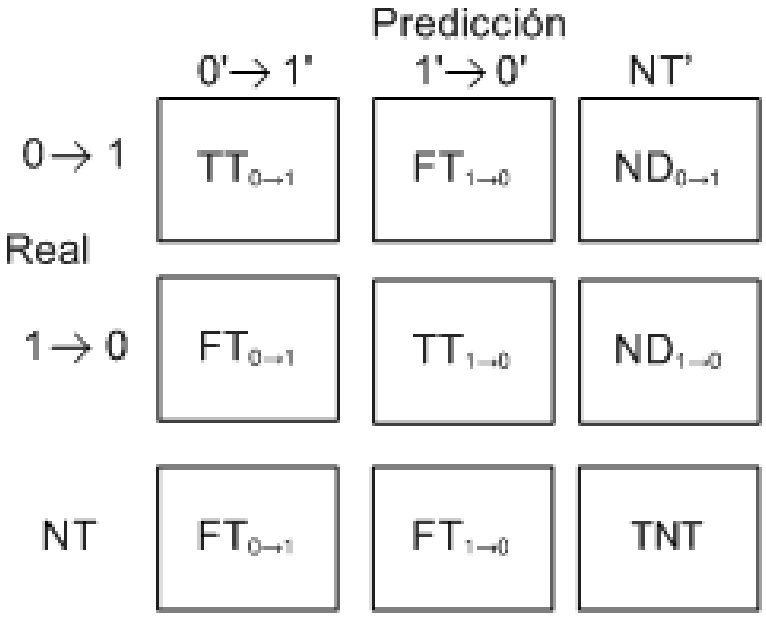


Figura B.1: Matriz de confusión de órden superior (simplificada)

Dónde:

- TT: transición existente y detectada.
- ND: transición existente y no detectada.
- FT: falsa transición.
- TN: transición no existente.
- TNT, verdadera no transicion

Para calcular dicha matriz se parte de dos secuencias de datos: la que contiene la clasificación real, la calculada, los estados; y el retraso máximo con el que se quiere trabajar. El valor del retraso máximo no resulta de utilidad en algoritmos que no tienen tendencia a retrasarse o adelatarse a un cambio de estado, de hecho su uso en estos casos sería contraproducente por que enmascararía algunos errores.

Con el valor del retraso máximo, se le permite al algortimo buscar los retrasos o adelantos en una ventana de tamaño $\pm retraso\,maximo$. La búsqueda comienza en el inicio de la transición real no detectada y termina al final de la ventana, no permitiendo que transiciones ya clasificadas, se puedan tomar como transiciones retrasadas o adelantadas.

Los valores del retraso máximo dependerán de las secuencias a considerar. Cuanto mayores sean dichos valores más posibilidades hay de que fallos al detectar una transición, se consideren un retraso o para valores pequeños, clasifiquen como error un retraso grande.

B.3.1. Entradas de la Función de Cálculo de Matrices de Confusión

La función presenta la siguiente llamada:

[confusion, porcentaje] = matriz_ confusion_ orden2(real, calculado, estados, retraso)

Dónde los parámetros especificados son:

- real: vector de datos correctamente clasificados.
- calculado: vector de datos calculados por el clasificador.
- estados: posibles estados que se pueden tomar.
- retraso: retraso o adelanto máximo que pueden tenerse en cuenta.

Ha sido probado correctamente con clasificadores de dos estados.

B.3.2. Salidas de la Función de Cálculo de Matrices de Confusión

La función:

[confusion, porcentaje] = matriz_ confusion_ orden2(real, calculado, estados, retraso)

Obtiene los resultados siguientes:

- *confusion*: matriz de confusion resultante.
- *calculado*: porcentajes de cada elemento de la matriz de confusión.

Ejemplo: Ejemplo de llamada a la función de cálculo de matrices de confusión:
calculado = [1 2 2 2 1 1 1...]
real= [1 1 2 2 2 1 1 ...]
estados=[1 2]

matriz_ confusion_ orden2(real, calculado, estados, 20)

B.4. Función score

Automatización de pruebas de evaluación de clasificadores y trayectorias para HMM y árboles. Situado en el subdirectorio *Ejemplos*.

Ejecuta una batería de tantas clasificaciones como ficheros reciba para un HMM por defecto o especificado; o para un árbol de clasificación.

[maC, confusion] = score(datos, tipo, varargin)

Siendo los posibles parámetros:

- *datos*: ficheros de datos con las trayectorias (array de *strings*).

- tipo $\begin{cases} 1 & HMM \\ 2 & Árbol \end{cases}$
  - Si es un HMM, se pueden especificar los valores de un HMM (π, A, μ y σ). De no especificarse, se utilizan los valores por defecto.
  - De querer ejecutar con un árbol, ha de introducirse un valor en el archivo de entrenamiento del árbol (llamado *ficheroARbol*, línea 36 del fichero *score.m*).

Salidas:

- *maC*: matriz de confusión (clásica).
- *confusion*: matriz de confusión de órden superior .

Ejemplo: Para árbol,

score(['ruta1', 'ruta2'], 2)

HMM por defecto,

score(['ruta1', 'ruta2'], 1)

HMM dado,
inicial = [1;0];
transiciones = [0.9950, 0.0050; 0.0006, 0.9934];
mu = [0.8590, 1.6242];
sigma(:,:,1)=0.0601;
sigma(:,:,2)=0.2383;

score(['ruta1', 'ruta2'], inicial, transiciones, mu, sigma)

B.5. Ejemplos

Además de todas las funciones indicadas en el manual mostrados, exisen ficheros con funciones de ejemplo ejecutables sin parámetro, que realizan clasificaciones y evaluaciones (situados en el subdirectorio *Ejemplos*):

- *hmm_ tiempo.m*: pequeño ejemplo de un HMM.
- *hmm_ bolas.m*: pequeño ejemplo de un HMM y clasificación por matriz de confusión clásica.
- *hmm_ casino.m*: ejemplo de funcionalidad utilizado para análisis de funcionalidad, sólo se ejecuta correctamente si se dispone del *Statistical Toolbox*.
- *hmm_ casino2.m*: similar al anterior.
- *hmm_ casino3.m*: similar al anterior, incorpora funcionalidad de la matriz de órden superior.

Apéndice C

Presupuesto

A continuación detallamos el presupuesto del proyecto, en el que se detallan las siguientes características:

1. Un total de 2 hombres mes (262,5 horas de trabajo) por parte de un Ingeniero Senior, encargado de la dirección y supervisión del proyecto.

2. Un total de 8,5 hombres mes (1115,625 horas de trabajo) por parte de un Ingeniero encargado del desarrollo y documentación del proyecto.

3. El número de horas por mes se establece en 131,25.

4. No existe una dedicación exclusiva de los integrantes del proyecto a la realización del mismo, por esta razón, algunas de las tareas que se especifican en la planificación, tienen duración en días, pero no existe un cálculo exacto de horas día.

5. Algunas de las tareas que se detallan en la planificación, concretamente las denominadas "*Descartar vías de Investigación Fuera del Presupuesto*", y también indicadas como líneas de investigación futura corresponden a tareas que en un principio pensamos podríamos desarrollar como parte del proyecto, pero debido a restricciones presupuestarias y de tiempo, no se han desarrollado.

6. Una vez comenzado el proyecto, en principio pensado para el desarrollo de una Herramienta de Clasificación de Series Temporales, decidimos abrir una nueva vía de investigación en la que desarrollamos una Herramienta de Evaluación de Clasificadores.

UNIVERSIDAD CARLOS III DE MADRID
Escuela Politécnica Superior

PRESUPUESTO DE PROYECTO

1.- Autor:
Gonzalo Pérez Verdú

2.- Departamento:
Grupo de inteligencia Artificial Aplicada

3.- Descripción del Proyecto:
- Titulo
- Duración (meses) **12**
Tasa de costes Indirectos: 20,00%

4.- Presupuesto total del Proyecto (valores en Euros):
Euros

5.- Desglose presupuestario (costes directos)

PERSONAL

Apellidos y nombre	N.I.F. (no rellenar - solo a titulo informativo)	Categoría	Dedicación (hombres mes) [a]	Coste hombre mes	Coste (Euro)
					0,00
Ingeniero Jefe		Ingeniero Senior	2	5.742,19	3.281,25
Gonzalo Pérez Verdú		Ingeniero	8,5	3.281,25	27.890,63
					0,00
					0,00
		Hombres mes	**10,5**	**Total**	31.171,88

[a] 1 Hombre mes = 131,25 horas. Máximo anual de dedicación de 12 hombres mes (1575 horas)
Máximo anual para PDI de la Universidad Carlos III de Madrid de 8,8 hombres mes (1.155 horas)

EQUIPOS

Descripción	Coste (Euro)	% Uso dedicado proyecto	Dedicación (meses)	Periodo de depreciación	Coste imputable [d]
Estación de trabajo	675,00	30	4	60	13,50
Portatil	1.200,00	20	3	60	12,00
Servidor	1.000,00	50	5	60	41,67
Impresora	250,00	100	1	60	4,17
				Total	71,33

[d] Fórmula de cálculo de la Amortización:

$$\frac{A}{B} \text{xCxD}$$

A = nº de meses desde la fecha de facturación en que el equipo es utilizado
B = periodo de depreciación (60 meses)
C = coste del equipo (sin IVA)
D = % del uso que se dedica al proyecto (habitualmente 100%)

SUBCONTRATACIÓN DE TAREAS

Descripción	Empresa	Coste imputable
Revisión de documentos	DAGM soluciones	100,00
Imprenta	S-Copy	50,00
	Total	150,00

OTROS COSTES DIRECTOS DEL PROYECTO[e]

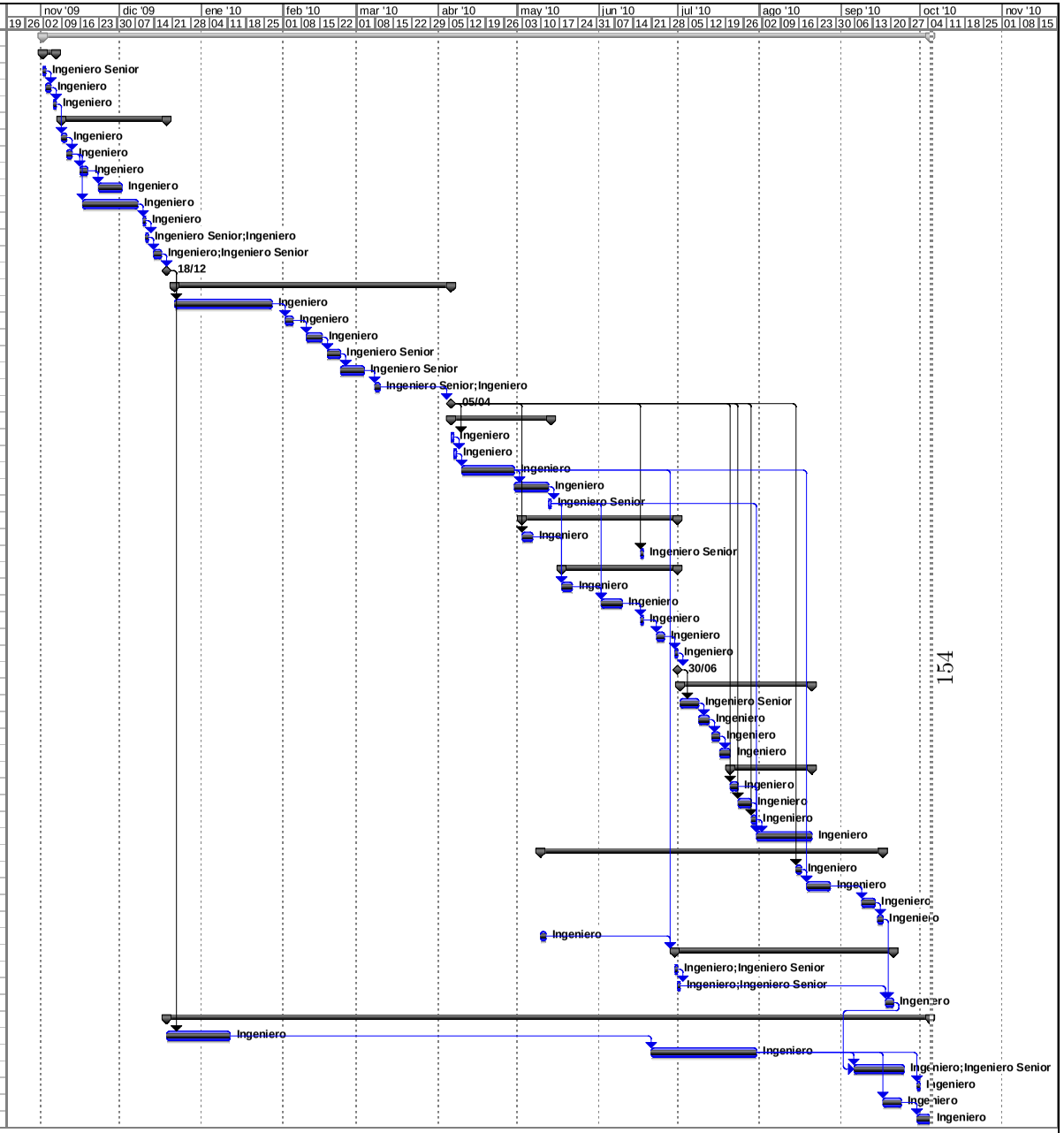
Descripción	Empresa	Costes imputable
Consumibles de oficina	La Rocha	25,00
Soporte de datos	La Rocha	35,00
Viajes y dietas	Repsol	520,00
	Total	580,00

[e] Este capítulo de gastos incluye todos los gastos no contemplados en los conceptos anteriores, por ejemplo: fungible, viajes y dietas, otros,…

6.- Resumen de costes

Presupuesto Costes Totales	**Presupuesto Costes Totales**
Personal	31.172
Amortización	71
Subcontratación de tareas	150
Costes de funcionamiento	580
Costes Indirectos	6.395
Total	38.368

Id	Nombre de tarea
0	**Herramientas de Segmentación y Evaluación de Series Temporales basadas en Modelos Ocultos de Markov**
1	**Ámbito**
2	Determinar el ámbito del proyecto
3	Definir recursos preliminares
4	Ámbito completado
5	**Análisis y requisitos del software**
6	Realizar análisis de necesidades
7	Borrador de las especificaciones preliminares del software
8	Desarrollar presupuesto preliminar
9	Revisar problemas planteados en el proyecto
10	Análisis de la herramienta de seguimiento de trayectorias
11	Calcular los tiempos y fechas de entrega
12	Obtener aprobaciones para continuar (concepto, fechas, presupuestos)
13	Análisis de posibles herramientas HMM disponibles en el mercado
14	Análisis completado
15	**Diseñar**
16	Estudio HMM
17	Estudio de funcionalidad de herramientas HMM
18	Desarrollar prototipos basados en herramientas HMM
19	Revisar herramientas y obtención de aprobación
20	Replanteamiento del problema de trayectorias
21	Obtener aprobación para continuar
22	Diseño completado
23	**Desarrollo Herramienta de Clasificación**
24	Revisar especificaciones de funcionamiento
25	Identificar parámetros de diseño
26	Desarrollar el código
27	Pruebas de los programadores (depuración primaria)
28	Desarrollo completado
29	**Pruebas**
30	Desarrollar pruebas
31	Planteamiento de necesidad de metodología de evaluación
32	**Pruebas y ejemplos simplificados**
33	Prueba de ejemplo discreta: problema dados
34	Desarrollo de documentación
35	Identificar problemas
36	Modificar código
37	Volver a probar el código modificado
38	Pruebas de unidades completadas
39	**Desarrollo Herramienta de Evaluación**
40	Planteamiento del problema
41	Búsqueda de cocumentación
42	Desarrollo de metodología
43	Desarrollo de herramienta
44	**Prueba de ejemplo discreta: problema dados**
45	Adaptar ejemplo dados a problema de Evaluación
46	Identificar posibles soluciones
47	Modificar código
48	Pruebas y ejemplos simplificados
49	**Prueba de ejemplo contínua: trayectorias**
50	Búsqueda de secuencias de prueba
51	Adaptar código
52	Descartar diseño de metodología de entrenamiento
53	Diseño y ejecución de pruebas
54	Pruebas completadas
55	**Descartar vías de investigación fuera de presupuesto**
56	Descartar entrenamiento
57	Descartar posiblidad de adaptar tamaño de ventana en residuos
58	Documentación
59	**Documentación**
60	Recopilar documentación generada a lo largo del estudio
61	Escritura de documentación
62	Revisión de documentación
63	Evaluar la información de las pruebas
64	Manuales de usuario y apéndices
65	Documentar conclusiones



Proyecto: Herramientas de Segmentac
Fecha: mar 05/10/10

Tarea | Progreso | Resumen | Tareas externas | Fecha límite
División | Hito | Resumen del proyecto | Hito externo

Bibliografía

[1] Lawrence R. Rabiner. "A tutorial on Hidden Markov Models and selected applications in speech recognition". Proceedings of the IEEE 77, vol 2, pp 257–286, Feb. 1989.

[2] George Box, Gwilym M. Jenkins, and Gregory C. Reinsel. "Time Series Analysis: Forecasting and Control", third edition. Prentice-Hall, 1994.

[3] Walter Zucchini and Ian MacDonald. "Hidden Markov Models for Time Series, An Introduction Using R", 2009.

[4] "Hidden Markov Models and the Baum–Welch Algorithm", IEEE Information Theory Society Newsletter, Dec. 2003.

[5] Jose Luis Guerrero. "Segmentación de Trayectorias Procedentes de Tráfico de Oportunidad", Proyecto de Fin de Carrera, Sept. 2008.

[6] "On Least Squares and Linear Combinations of Observations", Proceedings of the Royal Society of Edinburgh, vol. 55: 42–48, 1935.

[7] Cristina Esteve Elizalde, "Reconocimiento de locutor dependiente de texto mediante adaptación de modelos ocultos de Markov fonéticos", Proyecto de Fin de Carrera, Jul. 2007.

[8] Baldi, P.; Brunak, S.; Chauvin, Y.; Andersen, C. A. F.; Nielsen, H. "Assessing the accuracy of prediction algorithms for classification: an overview". Bioinformatics 2000, 16, 412–424, 2000.

[9] D. Peña Sánchez de Rivera, "Estadística: Modelos y Métodos", 1. Alianza Universidad Textos, Madrid, 1994.

[10] Anderson, Erling B. "Asymptotic Properties of Conditional Maximum Likelihood Estimators". Journal of the Royal Statistical Society B 32, 283-301.

[11] Wilcoxon, F. "Individual comparisons by ranking methods". Biometrics, 1, 80-83, 1945.

[12] Gini, Corrado. "Measurement of Inequality of Incomes". The Economic Journal (Blackwell Publishing) 31 (121): 124–126, 1921.

[13] Hand, D.J., & Till, R.J. "A simple generalization of the area under the ROC curve to multiple class classification problems". Machine Learning, 45, 171-186, 2001.

[14] www.mathworks.com/products/matlab/

[15] http://www.cs.ubc.ca/~murphyk/Software/HMM/hmm.html, Kevin Murphy, 1998.

[16] Kirkpatrick, S.; C. D. Gelatt, M. P. Vecchi. "Optimization by Simulated Annealing". Science. New Series 220 (4598): 671–680, May. 1983.